1994年7月1日発行（毎月1回1日発行）第13巻7号通巻147号 昭和58年11月2日第三種郵便物認可

PERSONAL COMPUTER MAGAZINE for MZ, X1, and X68000

特集 入門コンピュータミュージック
MUSIC SX-68K/ZPLK/内蔵音源の活用/波形メモリを使う
江口響子の実用講座/ビジネスショウ'94
付録5"2HDディスク CGA入門キット「GENIE」

7
1994

SOFT BANK オー/エックス
特別定価850円

ビデオグラフィックスの
世界へ。
資料請求券
X68030
Oh!X
7係
■お問い合わせは…シャープ株式会社
電子機器事業本部システム機器営業部 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)

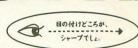
1,677万色対応、ビデオ映像を高画質・高速取り込み

テレビやビデオ、ビデオディスクなどの映像をX68シリーズやMacシリーズ※1の動画・静止画データとして高速取り込みが可能、いわば“ビデオスキャナ”とでも呼びたいビデオ入力ユニットです。1,677万色対応、最大640×480ドットの高解像度※2。動画・静止画の手軽なハンドリングが、新たなグラフィックシーンを創造します。

※1 MacintoshはIIシリーズ以降の機種に対応、ディスプレイ解像度が640×480ドットの場合、取り込み可能な範囲は、160×120ドット、320×240ドットのサイズになります。
※2 X68030/X68000シリーズでは、1,677万色はデータ作成のみに対応。表示は最大65,536色、解像度は512×512ドット。また、Macintoshは機種により表示色数が異なります。

アプリケーションツール「ライブスキャン」を標準装備

動画や静止画を簡単に保存できるアプリケーションソフト「ライブスキャン」※を標準装備。取り込んでいる映像を表示したり、残したいシーンを簡単に静止画保存したり、手軽な動画・静止画ハンドリングでパソコンの可能性をさらに広げます。X68030/X68000シリーズ用SX-WINDOW対応版とMacintoshシリーズ用QuickTime対応版の2種類を同梱しています。

※SX-WINDOW版はバージョン3.0以降(メモリー4MB以上)、QuickTime版はMacintosh漢字Talk7リリース7.1以上のシステムとQuickTime1.5以上(メモリー8MB以上)が必要です。

1,677万色対応の高速映像取り込み、動画・静止画の手軽なハンドリングが、新たなマルチメディアシーンを創造する。

SHARP INTELLIGENT VIDEO DIGITIZER CZ-6VS1　BUSY　POWER

■SCSIインターフェイス採用:パソコンの専用I/Oスロットを使わずに接続可能になり、汎用化を実現しました。またSCSI-2(FAST)インターフェイスの採用により、データ転送速度の高速化を図っています。X68030/X68000シリーズでは、SCSI-2(FAST)対応のハードディスクを接続することにより、パソコン本体を経由しないで、ハードディスクに直接、動画データをテンポラリデータとして記録することが可能です。パソコン本体のハードディスクへは、記録終了後に、テンポラリデータを変換し動画データとして保存できます。

※CZ-600C/601C/611C/602C/612C/652C/662C/603C/613C/653C/663Cに接続する場合は別売のSCSIインターフェイスボードCZ-6BS1ならびにSCSI変換ケーブルCZ-6CS1が必要です。※CZ-604C/623C/634C/644Cに接続する場合は、別売のSCSI変換ケーブルCZ-6CS1が必要です。
※Macintosh Power Bookシリーズに接続する場合は別売のSCSIケーブルなどが必要です。詳しくはMacintosh Power Bookシリーズの取扱説明書をご覧ください。

■高機能MPUを搭載:クロック周波数25MHzの32ビットMPU/MC68EC020を搭載、高速処理やパソコン本体の負担の軽減を実現します。

●MacはMacintoshの略称です。●Macintosh、Macintosh IIは、米国アップルコンピュータ社の登録商標です。●Power Bookは米国アップルコンピュータ社の商標です。●漢字Talk7はアップルコンピュータジャパン社の商標です。●QuickTimeは、米国アップルコンピュータ社の商標です。●価格には、消費税及び配送・設置・付帯工事費、使用済み商品の引き取り費等は含まれておりません。

ビデオ入力ユニット

CZ-6VS1

標準価格178,000円(税別)

SHARP わんさかフェア
■期間 '94 6/18(土)〜30(木) ■会場 九十九電機パソコン本店II 4F 東京都千代田区外神田1-9-9
■主催:シャープ株式会社 ■連絡先:シャープライブエレクトロニクス販売 中央情報システム営業部 ☎(03)5818-3268 担当 篠田
新型高性能スキャナ「JX-330X」とグラフィックを中心に68の世界を満喫ください。掘り出し物もアルヨ!

Oh!X

CONT

特集　入門コンピュータミュージック

麻雀航海記

The World of X68000Ⅱ

江口響子の実用講座

CGA入門キットGENIE

(で)のショートプロぱーてぃ

表紙絵：塚田　哲也

E　N　T　S

1994 JUL. 7

■広告目次

今も、先も楽しめる。いつも新展開の予感、SX-WINDOWのニューバージョン。

❶ マルチフォントエディタ編集例。文字ごとに文字種・文字の大きさの指定、装飾が可能。

❷ 用途に合わせてカスタマイズできる便利な「エディタ」モードの例。

❸ プリンタ印字例（カラー印刷は誤差分散により65,536色対応。）

❹ 「パターンエディタ」で作成したデータを背景に設定可能。

❺ 日本語フロントプロセッサASK68K Ver.3.0の辞書メンテナンスがウィンドウ上で可能。

❻ アイコンデータや背景データを作成する「パターンエディタ」。

❼ オリジナルに作成したアイコンパターンの例。

❽ 512×512ドットの範囲内で65,536色の表示が可能。

❾ さまざまなグラフィックフォーマットに対応。

❿ 任意のサイズに縮小・拡大表示が可能。

⓫ 異なる画像フォーマットへのコンバートが可能。

⓬ 別売のビデオ入力ユニットCZ-6VS1（標準価格178,000円）を介して映像ハンドリングが可能。

■実画面：1,024×1,024ドット、表示画面：768×512ドット

●画面は広告用に作成した、機能を説明するためのイメージ画面です。また、各種アイコンなどは、SX-WINDOW ver.3.1がもつ機能を使って作成したもので、標準装備のものとは異なるものもあります。
●本広告中のエディタで表示している文字のフォントはツァイト社の、「書体倶楽部」のフォントを使用しています。

●**インライン入力のサポート**：ASK68K Ver.3.0を利用したインライン入力をSX-WINDOWで実行可能。またシャーペン.Xをワープロとして利用できるよう、さまざまな機能が付加されています。

●**コンソールをサポート**：Human68kやX-BASICのコマンドをSX-WINDOWアプリケーションと同時にタイムシェアリングで実行できます。
（グラフィックを利用したものなど、SX-WINDOWと処理が重複するものは実行できません。）

●**多彩なプリンタに対応**：さまざまなSX-WINDOWアプリケーションで利用できるページプリンタドライバを標準装備。ESC/Page、LIPS III、PostScriptに対応したプリンタが利用できます。

SX-WINDOW ver3.1

CZ-296SS（130㎜FD）/CZ-296SSC（90㎜FD） 標準価格22,800円（税別）

●お問い合わせは…シャープ㈱電子機器事業本部（液映）システム機器推進プロジェクトチーム 〒162 東京都新宿区市谷八幡町8番地 ☎(03)3260-1161（大代表）へ シャープ株式会社

資料請求券 ペリフェラル Oh!X 7係

For X68030/X68000series APPLICATION SOFTWARE

◎パーソナルDTPをX68で

XDTP SX-68K

CZ-291BWD 6月発売予定

NEW

縦書きをはじめとした多彩なレイアウト機能で
パーソナルなデスクトップパブリッシングを実現するソフトです。
やさしい操作、豊富な編集機能、グラフィックウィンドウ対応、SX-WINDOWをすでに
ご利用になっている方なら、基本操作を新たに覚えることなく手軽にレイアウトが作成できます。

■豊富なテキスト編集機能：フォント種類、サイズ、文字種の変更はもちろん、上線、下線、網掛け、文字間隔の指定が文字ごとに設定可能。禁則、行間隔、タブ、インデント、マージンもパラグラフ(リターンコードまでの文字列)ごとに設定できます。また各テキストフレームごとに、フレーム形状、リンク状態(テキストの流し込み)、縦書き/横書き、回り込みの設定が可能。検索/置換も単純な文字列だけでなく、スタイル別に行うことができます。

■グラフィックウィンドウに対応：GRW.Xにも対応していますので、いろいろな形状でレイアウトしたグラフィックフレームのデータを65,536色の画像で確認しながらレイアウトできます。

■さまざまな画像フォーマットに対応：ビデオマネージャが対応している静止画フォーマットの他に、「PrintShop PRO-68K」、「CANVAS PRO-68K」、「GScriptファイル」の読み込みに対応しています。

●グラフィックフレーム、テキストフレームとは別に直線、矩形、楕円、多角形が作成できる独立した罫線機能●第1水準を収めた明朝体、ゴシック体のベジェーフォントファイルを標準装備●ページの移動や作成/削除がスピーディに行える独立したページウィンドウをサポート●ページプリンタドライバ(ESC/Page、LIPSIII)を付属、高解像度の美しい印字が可能。またSX-WINDOWが対応しているプリンタも使用可能。

※5MB以上の空きのあるハードディスクが必要です。 4MB、Ver.3.0

◎グラフィック感覚の楽譜入力をサポート

MUSIC SX-68K

CZ-274MWD 標準価格38,000円(税別)

NEW

MIDI、FM、ADPCMに対応した楽譜ワープロ&作曲演奏ソフトです。
自由なレイアウトでグラフィックを描くように楽譜入力、
全パートの同時入力や編集、自動伴奏機能、応用範囲を広げるデータ互換性。
多彩なプリンタ対応で美しい印刷も可能です。

■MIDI、FM、ADPCM対応：MIDI、FM、ADPCMを同時に発音できます。全ての音源を利用した場合、最大発音数は25まで設定可能です。

■全パートの同時入力：ピアノ譜、メロディ譜などの組み合わせで最大16パートまで編集可能。特定パートごとではなく全パートを画面に表示して編集できますので、直接画面上で曲の構成を考えながら作編曲できます。

■コード&リズムによる自動伴奏機能：メロディ上にコードネームとリズムパターンを入力するだけで、自動的に伴奏をつけることができます。

■優れたデータ互換性：「MUSIC PRO-68K」、「MUSIC PRO-68K[MIDI]」のデータファイルが利用できる他、OPM、MML、ZMSファイル形式でデータ出力が可能です。

■多彩なプリンタ対応：ページプリンタドライバ(ESC/Page、LIPSIII)を付属、高解像度の美しい印刷が可能です。
またSX-WINDOWが対応しているプリンタも利用できます。

4MB、Ver.3.0

その先のシーンへ。

●さらに実用的なウィンドウシステムへの進化

SX-WINDOW ver3.1 システムキット

CZ-296SS(130㎜FD)/CZ-296SSC(90㎜FD) 標準価格22,800円(税別) NEW

ASK68K Ver3.0を利用したインライン入力のサポート、Human68k/BASICコマンドをSX-WINDOWアプリケーションと同時にタイムシェアリングで実行できるコンソールのサポートをはじめ、シャーペン.Xをワープロとして利用できるよう機能アップ。また、さまざまなSX-WINDOWアプリケーションで利用できるページプリンタドライバを標準装備。ドローデータ(FSX)/フォントデータ(IFM)処理の高速化も実現しています。

※コンソールでは、SX-WINDOWと処理が重複するものは実行できません。 4MB

※既にSX-WINDOWをお持ちの方には有償バージョンアップサービスを行います。

●定評のGUI対応ウィンドウワープロ

EGWord SX-68K

CZ-271BWD 標準価格59,800円(税別) NEW

ウィンドウワープロとして評価の高いEGWordのSX-WINDOW対応版。キャラクタベースのワープロを超えたグラフィカルユーザーインターフェイス(GUI)による手軽なDTPソフトとしても優れた表現力を発揮します。定評ある日本語入力方式(EGConvert)によるインライン入力、さまざまなグラフィックデータ(GScript)やテキストデータの貼り込み、また文書互換を実現するEDF(Extended Document Format)形式をサポートしています。 4MB、ver.2.0

●SX-WINDOW開発支援ツール

SX-WINDOW 開発キット Workroom SX-68K

CZ-288LWD 標準価格39,800円(税別) NEW

SX-WINDOW用のソフト開発に必要なツールやサンプルプログラムを装備。プログラムの編集、リソースの作成、コンパイル、デバッグといった一連の作業をSX-WINDOW上で効率よく実行できます。初めてSX-WINDOW用のプログラムに挑戦する人にも、簡単に基本機能の理解が深まる33種(基礎編23種、応用編4種、実用編6種)のサンプルプログラム付き。

※ご使用に当ってはC compiler PRO-68K ver.2.1が必要です。 4MB、ver.2.0

●SX-WINDOW開発キットのサポートツール

開発キット用ツール集

CZ-289TWD 標準価格12,800円(税別) NEW

SX-WINDOW開発キットをさらに使いやすくするためのツールです。SXコールの簡易リファレンスを簡単に検索するインサイドSX、イベントの発生を常時監視・確認するイベントハンドラ、リアルタイムにメモリブロックの利用状況を表示するヒープビューアなど11種のツールが用意されています。 2MB、ver.2.0

●SX-WINDOW対応ドローイングツール

Easydraw SX-68K

CZ-264GWD 標準価格19,800円(税別)

イラスト、フローチャート、地図、見取り図など各種グラフィックが製図感覚で作成できます。作成したデータは他のSX-WINDOW対応アプリケーションでも利用でき、企画書などの作成をサポート。ページプリンタドライバも標準装備。 4MB、ver.3.0

●ウィンドウ対応グラフィックツール

Easypaint SX-68K

CZ-263GWD 標準価格12,800円(税別)

マウスによる簡単操作、65,536色中16色の多彩な表現、クリエイティブマインドに応えるウィンドウ対応ペイントツールです。同時に複数のウィンドウを開いて編集でき、各ウィンドウ間でのデータ交換もできます。 2MB、ver.1.1

●SX-WINDOWを楽しく使うためのアクセサリ集

SX-WINDOW デスクアクセサリ集

CZ-290TWD 標準価格14,800円(税別)

SX-WINDOWをさらに便利に楽しく使うためのデスクアクセサリ集です。スクリーンセーバ、スクラップブック、スケジューラ、アドレス帳、電子手帳通信ツール、パズルなど、12種の豊富なアクセサリが収められています。 2MB、ver.3.0

●マルチタスク機能をはじめ通信環境がさらに充実

Communication SX-68K

CZ-272CWD 標準価格19,800円(税別)

通信環境をさらに高めたウィンドウ対応の通信ソフトです。マルチタスク機能により他のアプリケーションを実行中でも簡単に通信が可能。自動ログイン機能やプログラム機能、など豊富な機能をサポートしています。 2MB、ver.1.1

●FM音源サウンドエディタ

SOUND SX-68K

CZ-275MWD 標準価格15,800円(税別)

他のミュージックソフトで演奏中の音色を、簡単に作成、変更できるマルチタスク機能、またエディット、イメージ、ウェーブの3つの編集/確認モードを装備。作成中の音色も50曲の自動演奏でリアルタイムに確認、編集できます。 2MB、ver.1.1

●SX-WINDOW対応になってさらにパワーアップ

倉庫番リベンジ SX-68K ユーザー逆襲編

CZ-293A(130mmFD)/CZ-293AWC(90mmFD) 各標準価格6,800円(税別)

倉庫番10年にわたるユーザーの投稿など、新作306面が目白押し。まさに倉庫番の最強版がSX-WINDOW上で楽しめます。AI機能やエディット機能、キャラクタ変更機能も装備。半年で解けたらあなたは天才?です。 2MB、ver.1.1

PRO-68K シリーズ

●X68030/X68000対応

CZ-295LSD 標準価格44,800円(税別)

C compiler PRO-68KのX68030/X68000対応版。MPU68030、MC68882の命令セットに対応したアセンブラ、デバッガ、ソースコードデバッガを付属。またHuman68k ver.3.0、ASK68K ver.3.0にも対応。新たにGPIBライブラリ、MC68882対応フロートライブラリを付属しています。

※メインメモリ2MB以上が必要です。
※C compiler PRO-68K/ver.2.0/ver.2.1をお持ちの方には有償グレードアップサービスを行います。

※(2MB,ver.1.1)の表示は、メインメモリ2MB以上、SX-WINDOW ver.1.1以上が必要であることを示します。

※発売予定のソフトの画面は実物とは異なる場合があります。

●EGWord、EGConvertは株式会社エルゴソフトの登録商標です。●ESC/Pageはセイコーエプソン株式会社の登録商標です。

●お問い合わせは…シャープ(株)電子機器事業本部(液映)システム機器推進プロジェクトチーム 〒162 東京都新宿区市谷八幡町8番地 ☎(03)3260-1161(大代表)へ シャープ株式会社

資料請求券

SHARP

高速、高解像度。

透過原稿・ADF対応型カラーイメージスキャナ、誕生。

アーケード版 ✚プラス スーパーファミコン版

PⅣ スーパーリアル麻雀

原画&設定資料集

●(株)セタ/監修
●田中 良/画

A3描き下ろしポスターつき

KAORI

YOU

ファン待望!!初の公式ブック登場

アニメを見るかのような滑らかなグラフィックでそれまでの単なる麻雀ゲームの常識を打ち破った「スーパーリアル麻雀」シリーズ。PⅡではじめてかわいい美少女キャラが登場して以来、現時点での最新作PⅣまで、常にトップの人気を集めてきた。本書はそのPⅣの魅力を全部公開するはじめての公式設定資料集だ。アニメーターとして活躍中で、シリーズすべてのキャラクターデザインおよび原画を担当してきた田中良氏の血と汗と努力が結集した原画の数々と描き下ろしイラストをとくとご覧ください。

定価2,000円

MANA

■定価は税込 ■お求め・ご予約はお近くの書店で

ソフトバンク株式会社/出版事業部 販売局：TEL.03-5642-8101

CGA入門キット「GENIE」
付録ディスクの内容

初めての人，過去に挫折した苦い経験がある人などにもCGAの楽しさを知ってもらうために，プロジェクトチームDōGAが熱意を込めて制作したCGA入門キット，その名も「GENIE」。パーツの組み合わせでお手軽にCGA制作が体験できるシステムです。ここでは，今回のディスクに収録されたデータを紹介しましょう。これを見ながら，あなたも何か作ってみてください。「GENIE」の使い方については連載のほうで解説します。

戦闘機：FT（FighTer）

FT01

FT02

FT03

FT04

FT05

FT06

FT07

FT08

FT09

戦艦：BS（BattleShip）

BS01

BS02

BS03

BS04

BS05

BS06

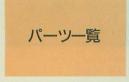
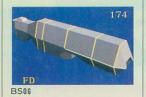
188
BS07

BS08

BS09

BS10

BS11

BS12

核：CR（CoRe）

CR01

CR02

CR03

CR04

CR05

CR06

CR07

CR08

CR09

CR10

円盤：DS（DiSk）

DS01

DS02

DS03

DS04

DS05

DS06

DS07

DS08

棒：PL（PoLe）
37
FD
PL01
56
PL02
62
FD
PL03
102
FD
PL04
142
PL05
147
PL06
150
PL07
176
PL08
199
PL09
218
PL10
平坦：FL（FLat）
45
FL01
94
FD
FL02
208
FL03
229
FD
FL04
261
FL05
横腹：SD（SiDe）
80
FD
SD01
82
SD02
101
FD
SD03
127
SD04
129
FD
SD05
140
SD06
150
SD07
150
SD08
翼：WG（WinG）
9
FD
WG01
25
FD
WG02
34
WG03
34
WG04
41
WG05
50
WG06
50
WG07
50
WG08
50
WG09
73
WG10
76
FD
WG11
132
FD
WG12
289
WG13
エンジン：EG（EnGine）
9
FD
EG01
9
EG02
14
FD
EG03
22
EG04
78
FD
EG05
85
EG06
98
EG07
99
EG08
102
EG09

121
EG10
164
TK06
62
BR02
47
SL05
120
CN05
121
FD
EG11
198
TK07
85
BR03
65
FD
SL06
126
CN06
操縦席：CP（CockPit）
連結：JT（JoinT）
156
EG12
26
CP01
91
FD
BR04
89
SL07
10
JT01
254
EG13
53
FD
CP02
95
FD
BR05
146
FD
SL08
10
FD
JT02
タンク：TK（TanK）
39
FD
TK01
77
FD
CP03
109
BR06
158
SL09
20
FD
JT03
楯：SL（ShieLd）
砲台：CN（CaNnon）
50
TK02
78
FD
CP04
16
FD
SL01
41
CN01
25
FD
JT04
87
TK03
83
CP05
22
SL02
60
FD
CN02
28
JT05
106
TK04
138
CP06
27
FD
SL03
75
FD
CN03
28
FD
JT06
艦橋：BR（BRidge）
120
FD
TK05
33
FD
BR01
28
SL04
102
FD
CN04
34
JT07

角錘，直方体など，単純な形状で，PR00に始まり，32個あります。ここでは全部を並べた画像を作成しましたので，それを掲載します。

江口響子の実用講座

夏のカードをフォトコラージュで

PhotoCDとMATIERを使えば絵が描けなくたってダイジョーブ！

マー坊，ウー助，スー太郎は3人兄弟。マー坊とウー助は，生まれたときから小さな村に住んでいますが，いちばん上のスー太郎は，弟2人のために町へ出稼ぎに出ています。

ウー助：スー太郎兄ちゃん，どーしてるかなー。久しぶりに手紙でも書こうか？
マー坊：夏のカードを，手作りで送ったら喜ぶよね，きっと。
ウー助：でも俺，絵なんて描けないぜ。
マー坊：ボクだって描けないけど……写真をハガキにしたらどうかな。去年の夏，みんなで海に行ったときのがあるじゃない。
ウー助：でも，ただの写真じゃつまんないよー。兄ちゃんだってもう見ているんだぜ。
マー坊：じゃ写真を組み合わせて，MATIERで加工したら？
ウー助：なるほど。で，文字も入れちゃう！
マー坊：写真をPhotoCDにしてから，CD-ROMドライブで読み込むと，スキャナよりもきれいで速いらしいよ。
ウー助：Macを使っている俺の友人が，CD-ROMドライブを持ってるぜ。
マー坊：たしか，X68000でもCD-ROMドライブが使えるようになるデバイスドライバ・ソフトが発売されてるはずだよ。
ウー助：じゃ，俺，探してくる……。

PhotoCD
これはフジカラーのもの。ほかにコダック，コニカのもある

X68000シリーズ用のCD-ROMデバイスドライバは，（株）計測技研から発売されています。
価格は，4,800円(税別)
(1)ISO9660アクセス対応ドライブ
　市販されているほとんどのCD-ROMドライブ
(2)CD-ROM　XA対応ドライブ(SX-PhotoGallery対応)
　東芝　XM-3301，XM-3401，およびその互換製品
　ロジテック　LCD-500
　松下電器　LK-RC533N25
　計測技研　KGU-XCD，KGU-XCD2
(3)オーディオコマンド対応ドライブ
　計測技研　KGU-XCD，KGU-XCD2
また，SX-PhotoGalleryは，PhotoCDをSX-WINDOW上で再現するソフトです。(CD-ROMデバイスドライバ付属)
価格は，15,000円(税別)
問い合わせ先　(株)計測技研　☎0286(22)9811

計測技研の
CD-ROMドライブ

このほかにも，いろいろなCD-ROMドライブが使える

ネガを選んでPhotoCDにする

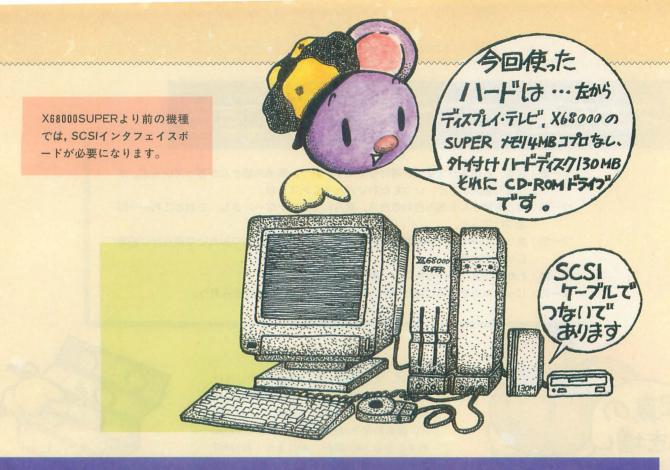
マー坊：まず，どのネガを使うか選ばなくちゃ。あっ，隣のお姉さんが写っているよ。ボクたち3人のこと，いつもかわいがってくれてたね。
ウー助：海とお姉さんを組み合わせたら，夏らしくていいなー。よし，これとこれ……印をつけてっと。
マー坊：あっ，ボールペンなんかじゃダメだよ……ダーマトグラフみたいな柔らかい鉛筆じゃなくっちゃ。
ウー助：わかったよ，わかったよ，これでOKだろ。
マー坊：じゃ，ネガをお店に持っていって，PhotoCDにしてもらおう。

注文は，大手のDPE店，量販店で受け付けてくれます。納期は，1週間から10日ぐらい。価格は，PhotoCDの原盤が1枚1,000円，技術料がその都度500円，入力が35mmのネガなら1枚100円，ポジなら1枚120円です。いちどきに入れると100枚まで入りますが，何回かに分けて入力すると，入る枚数は100枚以下になります。

ウー助：PhotoCDができたぞー。
マー坊：SCSIケーブルでCD-ROMドライブをつないで……これでよし。
ウー助：CD-ROMデバイスドライバをコピーして……CONFIG.SYSを書き換えるぜ。CD-ROMドライブのID番号は3にしたから，DEVICE=……CDDEV.SYS／ID3と書けばいいのか。
マー坊：PhotoCDを読み込むドライバと，SX-WINDOW上でPhotoCDが見られるPhotoGalleryをインストール。あとは，SX-WINDOWを起動するだけだね。

X68000SUPERより前の機種では，SCSIインタフェイスボードが必要になります。

今回使ったハードは…左から
ディスプレイ・テレビ，X68000のSUPER メモリ4MBコプロなし、外付けハードディスク130MB
それにCD-ROMドライブです。

SCSIケーブルでつないでありますす

画像をPhotoCDから SX-WINDOW上に取り出す

手順がちょっとめんどうだなあ

ウー助：CD-ROMドライブのアイコンをクリックと……あれ？　画像が出てこないぞ。
マー坊：IVM.XとGRW.Xを起動させて，グラフィックウィンドウを開いて……それからPhoto Galleryを起動する。その上にCD-ROMドライブのアイコンをドラッグするんだよ。
ウー助：あっ出てきた。へぇーきれいだなー。
マー坊：使う画像をだいたい決めておこうね。

◀SX-WINDOWの画面
手順を覚えにくいときは，使う順番にアイコンを並べておくと便利

画像をPICでセーブしてMATIERにもってくる

ウー助：ねぇ，使う画像を決めたのはいいけれど，どーやってMATIERにもってくるんだろう。
マー坊：まずクリップボードにコピーする……それからキャンバス，Xを起動して……ペーストする。ほらね。
ウー助：あっそうか。ここでAPICでPICファイルとしてセーブすればいいのか。

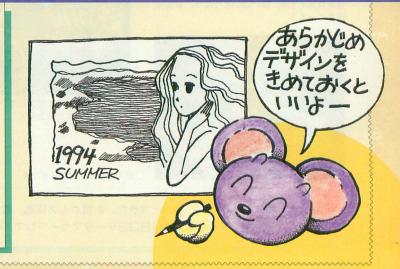
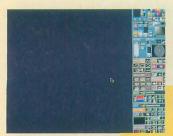
ウー助：よし、MATIERにもってくるぞー。ロードっと。あれっ？　画面が横長になっちゃった。
マー坊：PhotoCDの中の画像は，ピクセルの縦横比(アスペクト比)が1：1なんだ。MacintoshやPC-98なら問題ないんだけれどね。X68000はアスペクト比が3：4だから，こんな風にびろ〜んと伸びちゃうんだよ。だから，MATIERのトランスフォームで適当に修正するといいよ。
ウー助：ふーん，そうなのか。

デザインをおおまかに決める

ウー助：さ〜てと、フォトコラージュするぞー！
マー坊：どういうデザインにするかをおおまかに決めて，手順を考えてから制作にとりかかったほうがいいよ。そのほうが，無駄な作業が少なくなるからね。
ウー助：海を背景にお姉さんがふんわり白い線で浮き出ているのがいいなー。1994 SUMMERの文字も入れたい！
マー坊：う〜ん，写真を勝手に使って，お姉さん怒らないかな。ボク聞いてくるよ。

⋮

マー坊：出来上がったら，ポストカードくれればいいって。

MATIERでフォトコラージュ

□下地の色を決める

□海の風景をロード。トランスフォームしてレイアウトし，いらない部分を修正する

□風景のまわりをギザギザにトリミングする。いったん下地にマスクをかけ，マスク反転で，海の風景にマスクをかける。画面全体にノイズをかけて砂地の雰囲気を出す

□画面全体にデフューズを何回かかけて，パステルタッチにする。あとで合成する白い文字や線が目立つように，ダークツールを使って全体の色を少し暗くする。ここでいったん，裏画面の1にコピーする

□お姉さんの画像を読み込む。トランスフォームして修正する

□輪郭抽出をして，レイアウトする。裏画面の2にコピーする

□先ほどの海の画像を裏画面の1からコピーする

□お姉さんの画像を裏画面の2から表画面に減算コピーする。線が弱いようなら，もう一度コピーする

□文字を入れ，その部分にデフューズをかけて砂目状にして出来上がり

□髪の毛などを加筆修正する

MATIERのいろいろな使い方は，CG専門誌「PIXEL」の1994年2～6月号で，長谷川一光先生が詳しく解説しています。

MATIER　ver.2.0
X68000用　5″2HD 2枚組
39,800円(税別)
サンワード
☎044(855)4335

ウー助：ふう〜，やっとできた。お姉さんの輪郭，筆でさっと描いたみたいだ。
マー坊：ほんと。あとはRGBデータでセーブして，いよいよポストカードの発注だね。

ラベル記入例

SEA．RGB
512×512ピクセル
アスペクト比３：４
村のマー坊，ウー助
TEL 9999-88-7777

ファイルは必ず
大文字でセーブ

ウー助：データをセーブしたディスクは，どうすればいいのかなぁ〜。
マー坊：ラベルに，ファイル名，ピクセル数，アスペクト比，氏名，電話番号をきちんと記入した？
ウー助：したぜ。で，PhotoCDみたいに，DPE店に持っていけばいいんだろ？
マー坊：CG出力は，ちょっと特殊だから，どこでもOKというわけにはいかないんだ。（株）フジカラーサービスの場合は，東京と大阪のプロフェッショナル営業所で受けつけてくれる。でも，あらかじめCGの担当者に問い合わせたほうがいいよ。郵送でも受けつけてくれるらしいけど……。でね，RGBデータから直接ネガに焼くこともできるけど，色が思ったとおりに出ているか不安だったら，まずポジに焼いてチェックしてから，ネガに起こすといいね。

CGポジ出力サービスの価格は，（株）フジカラーサービスでは3,000円（税別）です。
また，ポジからネガへ起こすには，1,000円（税別）です。

ディスクのデータからポストカードへ

ウー助：ポジができたぞ〜。あれっ？　画面で見た色よりちょっと暗いみたい。
マー坊：画面の色は輝度が加わっているから，その分明るく見えるんだ。気になるようだったら，作るときに少し明るめにするといいね。
ウー助：それに，画像のまわりに黒い枠ができてるよ。
マー坊：ああ，それはね……35㎜は普通カメラで撮るサイズでしょ。でもCGの場合は，カメラのときと違ってどうしてもまわりが余ってしまうんだ。余りの部分が黒くなるんだよ。だから，トリミングをしないと，黒枠のついた写真ハガキになってしまうこともあるんだ。CGを写真ハガキにするなら，トリミング指定のできるプロポストカードのほうが安心だよ。

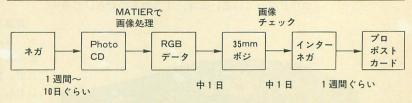
ポジ

インターネガ

制作の流れ

プロポストカード　全面タイプ

枚数	価格（１枚）
10〜29枚	90円
30枚以上	80円

プロポストカード技術料（トリミング指定などを含む）1,200円
（株）フジカラーサービス

ウー助：やったー。ついにできたー！
マー坊：きれいだねー。作業時間そのものは，大したことなかったけど，出来上がるまで時間がかかったね。
ウー助：はやめに作りはじめて，よかったなー。
マー坊：スー太郎兄ちゃん喜ぶよ，きっと。

PhotoCD，CG出力，プロポストカードの問い合わせ先

（株）フジカラーサービス　プロフェッショナル営業所　映像システム課
〒141　東京都品川区西五反田３-６-30
☎03(3493)6811　担当　富岡さんまで

［第38回］
星籠

江口　響子

響子inCGわ〜るど

昼と夜が交差する，たそがれどき。

文語の「誰そ彼は(たそかれは)」に由来しているといいます。薄暗がりに誰かがいるのはわかるけれど，誰だかはわからない，そんな時間に何も考えずぼんやりしていると，心の闇にストーリーが突然浮かんでくることがあります。見極めようとして近づいたとたん，風に吹かれた砂の城のように崩れ，跡形もなくなる……こんな風にして一瞬で消え去ってしまった，たくさんのストーリーの輪郭や断片を記したノートから，7月らしいものを選んでみました。

七夕の日の夕暮れ。公園のベンチ。

暗くなって字が読みづらくなったので，本を閉じ，そろそろ帰ろうかと考えていると，

「一緒に，星を拾いに行かない？　星籠を持っているの」

と声がした。

振り向くと，目の前にショートヘアの小さな女の子が立っている。年は，7つか8つぐらいだろうか。差し出した右手には，不思議な形の籠がぶら下がっていた。

「七夕の夜はね，星がいっぱい落ちてくるの知ってた？　織姫が天の川を渡って，彦星に会いにいくでしょ。そのときにね，天の川から星があふれてこぼれるの」

天の川に流れ星だって!?

東京の夜は，星は数えるほどしか見えない。正月やお盆の車の少ないときでさえ，たかが知れている。夏のはじめのいま時分なら，夏の大三角形，宵の明星の金星，北極星，さそり座，北斗七星といくつかの星ぐらいだ。天の川や流れ星なんて，都心ではとても見ることはできない。

「星を見るんじゃなくて……拾うの。落ちた星はじきに消えてしまうけれど，そのまえに星籠に入れれば，一晩は輝いているのよ。流れ星だから，願い事もできるの」

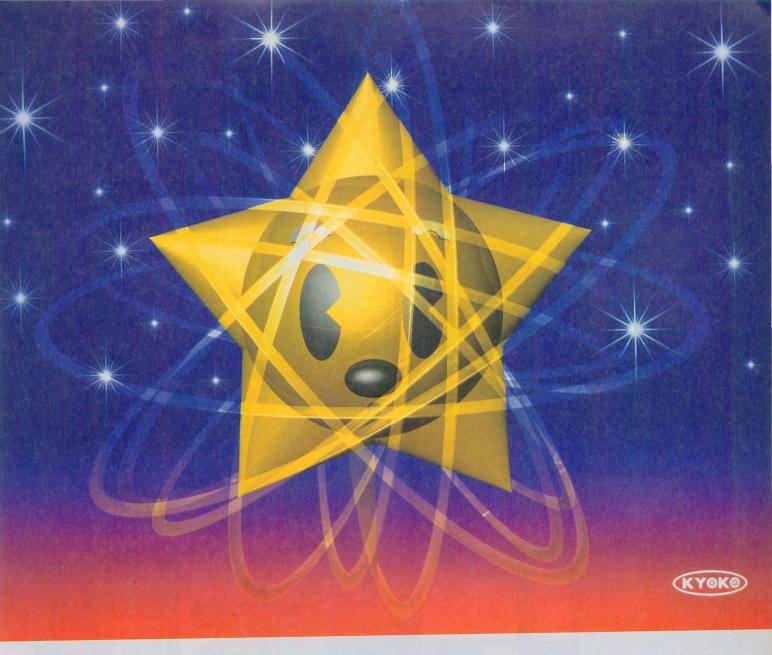
彼女は相手の心が読めるのだろうか，こういったのだった。

見渡すと，星はあちこちに落ちていた。電話ボックスの陰，歩道の脇，電線にひっかかっているのもある。しばらくは，きらきらと輝いているのだが，数分すると光は弱くなって消えてしまう。しかし，道行く人は誰も気がつかなかった。まさか星が落ちているとは思わないのだろう。

ひとつの星が足元に落ちた。

「さ，はやく」

と彼女が籠を開けてくれたので，すぐに拾い上げて中に入れた。

「星って，通ったあとに軌跡ができるでしょ。この籠はね，星の軌跡で編んであるの。だから，その中にいると星は，空にいるのとおんなじ気分になって安心するの。一晩は輝きつづけるのよ。でも，夜が明けると消えてしまうから，願い事をす

るなら今晩のうちにね」

彼女と別れて家に帰った。部屋を暗くして，机の上に星籠を置いた。何を願えばよいのかと考えた。どういうことが心からの願い事というのだろうと思った。

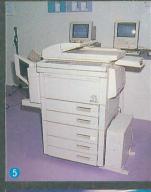
星の光は，強くなったり弱くなったりしている。心臓の鼓動のようだった。黄色くてやわらかな光が籠からこぼれている。それをじっと見つめていると，頭のなかがぼうっとしてくるのだった。

朝になっていた。星は消えてしまった。星籠だけがテーブルの上に残っており，影が窓からの光をあびて，細く長く伸びている。そして，気がついた。一番の願い事は，少女が一体誰なのか知りたかったのだと。

今回のCGデータ

1920×1536ピクセル
1670万色フルカラーを4×5ポジで出力
使用ソフトはC-TRACE
総物体数86(物体数57，論理演算29)平行光線1
MATIERで作成した背景を，αチャネルで合成

ビジネスショウ'94

BUSINESS SHOW '94

5月17日から20日までの4日間，東京晴海の国際見本市会場でビジネスショウが開催された。

シャープではNEWTONの日本版GALILEOを発表。注目を集めていた。

松下のWOODYは「テレビパソコン」という懐かしい響きのあるマシンだ。WINDOWSマシンを核にCD-ROMメディアとテレビを統合したマシンを提唱したものだ。特にCD-ROMメディアに力を入れているようでCDGカラオケやビデオCDなどにも対応している。もちろん昔のテレビパソコン(CF-3000だっけ？)の面影は微塵もない。

同様のコンセプトによるIBMの新PS/Vも発表されていた。そのほかOSの統一を狙うIBMのWorkplaceのDOS版も出展。

Macintosh関係はPowerアプリを一堂に集めるなど実に元気だ。

周辺機器関係ではソニードライブを使った低性能ながら低価格なMOドライブやエプソンの720dpiのカラーインクジェットプリンタ，ついに量産段階に入った大型FLCディスプレイなどが注目されるところだ。画質はいまひとつだが，15インチが最小級という大画面は魅力だ。

❶これがGALILEOだ ❷ICMの低価格MOドライブ ❸専用紙使用時720dpiの超高解像カラープリンタ ❹アナログなので画質は最高。低価格熱転写プリンタ（ページ型） ❺最近のコピーはファクスとページプリンタを兼ねる ❻15型FLCディスプレイ ❼驚異の自動ページめくりコピー機 ❽これが松下のWOODYだ ❾WorkplaceはOSを統一するか？ ❿DTPの極み？ 330線対応の印刷システム ⓫ビジネスにはリクライゼーションも大切かも

SOFTWARE INFORMATION

現在，読者の期待度No.1のゲーム「餓狼伝説SPECIAL」ですが，とうとうX68000版の画面をお見せできることになりました。完成まではまだ時間がかかるようで，発売予定は夏。あと少しの辛抱です。

餓狼伝説SPECIAL

あの「餓狼伝説２」に引き続き，移植が進められている「餓狼伝説SPECIAL」だが，開発途中バージョンが編集部に届いた。これで，「餓狼伝説」シリーズは，すべて移植されたってことだね。NEO•GEO版の発売からはだいぶたったが，ちゃんと移植されるなら，それだけでも評価しちゃうかな。

ただ，「餓狼伝説２」が10MHzであの速度だったから，スピードアップした「餓狼伝説SPECIAL」の移植はちょっと気になってしまう。このへんについてはまだわからないので，製品版に期待しよう。

それでも，あのミスター当て身のギース様が使えるとなっては，ゲーマーの貴方はもう黙っていられない。タンフールーの旋風剛拳も「餓狼伝説SPECIAL」で復帰。リズミカルなダックキングも「打倒，テリー」で復活！　前作を遥かに凌いだこのゲーム。ハードディスク容量もかなり必要だろうから，この際MOでも購入して，待つことにしようか。

個人的にはリチャードマイヤーがいないのが残念だけど，ファン待望の１本となることは間違いないだろう。（瀧）

X68000用　5″2HD版　価格未定
魔法株式会社　☎078(261)2790

期待が急騰の２作

		（前回順位）
1.	SX-WINDOW ver3.1	1
2.	餓狼伝説SPECIAL	2
3.	XDTP	−
4.	スタークルーザーII	8
5.	Mr.DO!/Mr.DO! VS UNICORNS	−
	サムライスピリッツ	10
7.	魔法大作戦	6
8.	龍虎の拳	10
9.	MUSIC SX-68K	3
10.	The World of X68000 II	−

６月号の読者アンケートはがきによる「期待の新作ソフト」です。

１，２位は先月号から変化なしという結果で，この２本は３位以下を大きく引き離して期待を集めています。１位の「SX-WINDOW ver3.1」はすでに店頭に並んでいます。「餓狼伝説SPECIAL」のほうは発売日は未定ですが，上でも紹介しているように開発は順調のようです。どちらも，前バージョン，シリーズ前作の好評が大きく影響していると思われます。

新顔は３本が登場しました。いずれも先月，先々月号あたりから情報が出始めたものです。「XDTP」「Mr.DO!/Mr.DO! VS UNICORNS」については今月号でも新しい情報を提供できないのが残念ですが，来月号では何かお届けできるでしょうか。「The World of X68000 II」は，完成直前のサンプル版にての紹介記事を掲載しましたので読んでくださいね。この号の発売前には製品が書店に並んでいる予定です。

最後に，残念なお知らせがひとつあります。開発着手の発表以来多くの期待を寄せられていた「龍虎の拳」ですが，諸般の事情により開発中止が決定となりました。ファンだけではなく，いままで開発に尽力し続けてこられたスタッフの方々にとっても残念なことと思います。ユーザーとしては，次回作に期待したいですね。

レッスルエンジェルス3

先月号でも紹介した「レッスルエンジェルス3」だが，移植は順調のようだ。これまでの2作品との大きな違いはプロレス団体の経営にあるが，今回はそれについて少し触れておこう。

まず，団体だが，最初からメンバーが決まっているわけではない。これは，他団体の現役選手や引退選手，新人などをスカウトしていくのだ。前作，前々作でお気に入りの選手がいる場合には，ぜひスカウトしよう。

さて，どのようにプロレス団体を経営していくのか？　それは興行である。興行とは地方の体育館などでよく行われるアレである。実際に見たことがなくとも，当日には宣伝カーなんかで派手に宣伝されているので，きっとご存じだろう。利益を得れば，有力な選手をスカウトするなど，さらに団体を強化できる。

ゲームモードは，団体経営モード，エキシビションマッチ，団体対抗戦の3種類。団体対抗戦では，団体経営モードで育てたディスクをもちよることで，ほかのプレイヤーの団体と対抗戦を行うことができる。

発売日は7月31日の予定。

X68000用　3.5/5″2HD版　5,800円(税込)
TAKERU　☎052(824)2493

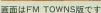
クイーン・オブ・デュエリスト外伝＋α

アグミックスから発売されている「クイーン・オブ・デュエリスト」シリーズ。美少女バトルゲームだが，漫画家がキャラクターをひとりずつデザインしているので，いろいろな個性が楽しめる。特にその作家のファンの人には嬉しい企画である。

ゲーム内容は，PC-98版，FM TOWNS版と発売を重ねるたびにバージョンアップが重ねられ，とうとう格闘ゲームの本命(？)X68000版の開発が開始された。背景はすべて描き直し，アニメーションを滑らかにするためにキャラクターパターンも増やし，思考ルーチンも強化するとのことなので，他機種でのプレイを経験済みの人も楽しみに待っていよう。

美少女もので2つのバージョンがあるが，TAKERUでは誰でも購入できる一般バージョンのほうが発売される。18禁バージョンはパッケージ販売で，価格が9,800円(税別)。

X68000用　3.5/5″2HD版　5,800円(税込)
TAKERU　☎052(824)2493

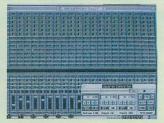
画面はFM TOWNS版です

Mu-1GS

最初に広告が出て以来，長らく待たされた音楽ツール「Mu-1 GS」が，ようやく発売されることになった。今回の主なバージョンアップ点は，GS/GM音源への対応，スタンダードMIDIファイルのコンバータのデバッグ，そして，ステップエディタ機能の強化などだ。

なかでもステップエディタ機能の強化は最も充実していて，ランダムに音のばらつきを与えて手弾きのリアルさをシミュレートする機能や，変数などを含む置換機能などがある。

当然のことながら，これまで好評だったリアルタイム録音機能も健在なので，適当に録音したあとで，うまく弾けなかった部分だけをステップエディタで手直しすることが可能である。

そのほかの機能はそのままだが，MT-32系列のLA音源のエディタは削除されているので，対象はGS/GM音源ユーザーに絞られている。(瀧)

X68000用　5″2HD版　28,000円(税別)
サンワード　☎044(855)4335

宝魔ハンターライム12

当初は6話で完結予定だったのを，倍に増やして続けられた人気シリーズも，とうとうこれが最終回。ディスク5枚組と豪華版で，グラフィックもアニメーションも充実，ファンは大満足だろう。

そもそも，ライムたちが人間界にやってきたのは，妖怪となった魔宝玉を探し出して，人間たちに迷惑をかける妖怪を退治するのが目的だった。そして，魔宝玉を回収すれば帰れるはず……なんだけど，そんなことはもうどうでもいいみたいだぞ。こっちでの生活もなかなか楽しいし，仲よしの友達もいるしね。

で，のんびり平和にすごすライムたち。だけど，ちょっと何か忘れてはいないかい……？ そう，第6話で戦ったアイツである。まだ勝負はついていないんじゃなかったっけ……。

最終話だから，当然ながらそのボスキャラが登場する。手強い相手にふいを突かれたライムちゃんは絶体絶命！ 動転するプギ。バース，ココナは？ そして，みづきちゃんは？

さあ，感動のエンディングをお楽しみに。

X68000用　3.5/5"2HD版　1,500円(税込)
TAKERU　☎052(824)2493

画面はPC-9801版です

XCASE

これまでX68000には，サードパーティから発売された開発ツールはほとんどなかった。しかし，このほどBeシステムより，純正品での不満点を補足し，ソフトウェア開発を効率化するためのCASEツールの発売が予定されている。

この「XCASE」は，PADチャート(構造化フローチャート)の形でプログラムを入力するもので，図形や表を使って可能な限りのプログラミングの自動化を図る。作成されたプログラムはコンバート機能により，CまたはアセンブラのソースファイルにСに出力される。

そのほか，データやドキュメントなどを一括管理するデータベース機能も備え，書式の統一や，ソフトウェアとドキュメントとの対応をチェックするなど，プログラミングだけではなく全般的な開発効率の向上を目指している。

Beシステム　☎0492(94)7966
X68000用　5"2HD版　19,800円(税込)

発売中のソフト

★雀神クエスト　SPS　5/19
X68000用　5"2HD版　7,800円(税別)
★SX-WINDOW ver3.1システムキット　シャープ　6/上
X68000用　3.5/5"2HD版　22,800円(税別)
★The World of X68000 II　電波新聞社　6/上
X68000用　5"2HD版　4,900円(税込)
★宝魔ハンターライム11　TAKERU　6/10
X68000用　3.5/5"2HD版　1,500円(税込)
★麻雀航海記　TAKERU　6/15
X68000用　3.5/5"2HD版　5 800円(税込)

新作情報

★宝魔ハンターライム12　TAKERU　7/10
X68000用　3.5/5"2HD版　1 500円(税込)
★レッスルエンジェルス3　TAKERU　7/31
X68000用　3.5/5"2HD版　5 800円(税込)
★餓狼伝説SPECIAL　魔法株式会社　7/下
X68000用　5"2HD版　価格未定
★Mu-1 GS　サンワード　5/未
X68000用　5"2HD版　28,000円(税別)
★ロボスポーツ　イマジニア
X68000用　5"2HD版　価格未定
★Traum　象スタジオ
X68000用　5"2HD版　価格未定
★鮫！鮫！鮫！　KANEKO
X68000用　5"2HD版　価格未定
★達人　KANEKO
X68000用　5"2HD版　価格未定
★エアバスター　KANEKO
X68000用　5"2HD版　価格未定
★サバッシュII　ホブコムソフト/グローディア
X68000用　5"2HD版　価格未定
★麻雀悟空・天竺への道　シャノアール
X68000用　5"2HD版　9,800円(税別)
★スタークルーザーII　アルシスソフトウェア
X68000用　5"2HD版　価格未定
★魔法大作戦　EAビクター
X68000用　5"2HD版　価格未定
★地球防衛MIRACLE FORCE　カスタム
X68000用　5"2HD　価格未定
★XDTP SX-68K　シャープ　6/未
X68000用　3.5/5"2HD版　価格未定
★Mr.DO!/Mr.DO VS UNICORNS　電波新聞社　6/下
X68000用　5"2HD版　5,300円(税別)
★クイーン・オブ・デュエリスト外伝＋α　TAKERU　8/未
X68000用　3.5/5"2HD版　5,800円(税別)
★X CASE　Beシステム
X68000用　5"2HD版　19,800円(税別)

魔法株式会社より発売が予定されていました「龍虎の拳」は，残念ながら開発中止となりました。

3つの味で麻雀三昧

Sudo Yoshimasa
須藤 芳政

パソコンで遊ぶ麻雀って「麻雀する」のもさることながら，もうひとつのお楽しみもとっても重要。今月は，それぞれ違ったお楽しみ要素をもった3本の新作ゲームをまとめて紹介しちゃいます。中華に飽きたら和風もイイかな!?

近頃，友人たちがあちこちへバラバラになってしまったため，麻雀を打つ機会がなくなった。苦労してゴミ捨て場から拾ってきたコタツ板の立場はいったいどうなってしまうのだろうか？

私と同じような境遇の方々，我々には麻雀マシンX68000があるではないですか。さあ，真夏の太陽が照りつけるなか，カーテンを閉め切った暗い部屋で，「おいおい，暗カンにウラが4つも乗って親倍やっちったよ。待ちがラス牌だったからさー，まさかツモると思わなかったんだよねー」と，誰に告げることもない孤独な勝利をディスプレイに向かって熱く語るのだ。

今回は「スーパーリアル麻雀PⅣ」のバトルモードと「麻雀航海記」「雀神クエスト」を3つまとめてご紹介しよう。

一発勝負のバトルモード

パソコン版「スーパーリアル麻雀PⅣ」には，アーケードからそのまま移植された「PⅣモード」のほかに，「PⅡ」のショーコ，「PⅢ」のカスミとミキ，そして「PⅣ」の3姉妹の計6人によって競われる「麻雀バトルモード」がある。

●スーパーリアル麻雀PⅣ
X68000用 5″2HD版 12,800円(税別)
ビング ☎03(3492)1079
●麻雀航海記
X68000用 3.5/5″2HD版 5,800円(税込)
TAKERU ☎052(824)2493
●雀神クエスト
X68000用 5″2HD版 7,800円(税別)
SPS ☎0245(45)5777

3等身キャラもなかなか可愛いぞ

プレイヤーは6人のうちから好きな女の子を選び，彼女に代わって対戦相手と1対1で麻雀を打つ。早い話が貴方は「代打ち」である。しかし，負けたからといって「この落とし前，どないしてけつかるんや！」と怖いお兄さんにどつかれる危険はないので，寝っ転がるなり，逆立ちするなり，豆腐の角に頭をぶつけるなりして楽な姿勢で挑もう。あ，やっぱり食べ物を粗末にしてはいけないということで豆腐は禁止する。

さて，私は誰を選びましょうかね。えーと，ショーコさんに決定！ 理由はほかの5人がまともな名前なのに対して，彼女だけが「ロンよりショーコ」という人生を投げたような名前であることに同情したからだ。これなら「悪魔ちゃん」のほうがよっぽどマシだぞ。

さて，勝負は一度きり。つまり，どちらか先に和了ったほうが勝ち。流局時のテンパイ，ノーテンは関係なし！ いくら「どひゃー！ 生まれて初めて九連宝燈テンパったぞ！ いままで辛いことや悲しいこともあったけど，生きててよかった！ 故郷のお父さーんお母さーん！」などとエキサイトしたところで，相手に「タンヤオのみ」で和了られてしまえばそれまでだ。即リーチの早和了りしか手はない。

やった！ 苦闘の末にショーコさんで5人勝ち抜いたぞ。優勝者には「ミス・スーパーリアル麻雀」の栄冠と，どんな夢でも叶えてもらえる特典があるとか。わーい！ どんなお願いしようかな？ よし！ 私の口をヨガファイアーが吐ける構造にしてもらおう。何!? 叶えるのはショーコさんの夢だけ？ がっくり。

船酔いしそうな長旅だ

「麻雀航海記」の舞台は中世の時代。アウストラロピテクスが全盛だった頃でないことは，中学生のお子様でもわかるだろう。

主人公である貴方は，没落商人の3代目として生まれ，運よく天涯孤独の身となり遺産相続で人並みの財産を手に入れた。

しかも折よく，国王から「この10年で成功した者に我が娘をくれてやる」とのなんともすてきなおふれが！ 逆玉のチャーンス！ しかし，何をすれば「成功を収めた」ことになるのか見当がつかない貴方は，ロッコのもとへと足を運ぶ。ロッコは，元○ベッカのヴォーカルでも，貝を腹の上で割って食べるラブリーな動物でもなく，町で評判の物知り爺さんだ。爺さんの話によると，我が国との貿易を渋っている7つの国があり，その各国との貿易ルートの開拓こそが「成功」への手段だというのだ。

貴方はさっそく航海に必要な食料，イカサマアイテム(麻雀の勝負の際にイカサマをはたらくためのアイテム)を買い込み，ロ

ショーコ，その悔しさをバネにがんばるんだ！

ッコと共に愛舟「これじゃあ沈んで当たり前だ」号で大海原へ旅立った。

貿易開拓の条件は，麻雀２番勝負で勝つこと。相手を２回ハコテンにすれば貿易が許されて，その地の特産物を買うことができる。

手始めは近場の北アメリカでタバコを買いにいってみよう！　……と思ったが，私はタバコを吸わないので日本へ出向くことにした。さあ，勝負勝負！　リメンバー・パールハーバーじゃ！　……しかし，勢いだけで麻雀には勝てなかった。何度もしつこく挑戦したにもかかわらず，結局和了れたのはタンヤオ１回のみ。相手はダブルリーチの連続で，しまいには人和(レンホー)を和了られる始末で，士気の表示が「最悪」になってしまった。

「クッソー！　こんなのイカサマやんか！お前のような奴は超必殺フライングクロスチョップをお見舞いしてやる！」

「ひいー！　おたすけー！」

……といきたいところだが，「フライングクロスチョップ」のコマンドなどないので母国へおとなしく帰るしか道はなかった。

「ついてないなー」

なんて思いながら母国へ向かう途中で，舟が停止した。水夫達が反乱を起こしたというのだ！

「我々はー！　雇用改善を要求するー！」

彼らは1000Gの金を要求してきた。雇われの身で何をいうか！　のび太より生意気だぞ！　スケさん，カクさん，こらしめておやりなさい！　ジャーン，ジャジャジャーン……しかし，周囲を見渡すと私の味方はロッコ爺さんしかいなかった。爺さんともども鮫のエサになってゲームオーバー！

なんということだ！　私としたことがとんだミステイクをやらかしてしまったぞ。おそらく，水夫たちに休暇を与えなかったのが原因と思われる。今度からは気をつけよう。

さて，日本は諦めて北アメリカへ向かった私はインディオと麻雀対決。ちょっとてこずったが，一度和了るとツキがこちらへ傾いたのか，あとはストレートに和了りまくって，とうとうタバコを買いつけることに成功した。買ったタバコは母国へ持ち帰って買い取ってもらう。その差額が儲けというわけだ。このようにして「成功」へ向けてひたすら航海を繰り返す。

ところでこのゲーム，プレイしていてたびたび，

「相手はツミコミでもやってるんじゃないのか？」

と思ってしまう。リーチが異様に早いのだ。こんな奴らからどうやって和了るのかと考えたが，前に述べたイカサマアイテムが必要なのではないだろうか？　きっと，そうに違いない。でも高くてなかなか買えましぇん……。健闘を祈る。

民族色豊かなグラフィックなのだ

アイテムは高価だけど手に入れたいなぁ

水夫たちのごきげんとりも大事だぞ

脳を高速回転せよ!

前々から発売が予定されていた「マージャンクエスト」だが，「雀神クエスト」と改名されていよいよ発売された。

全体的に「○ラ○ンクエスト」を意識した印象を受けるが，マップ上を自由に動き回れるというわけではなく，物語の進行に合わせて主人公が移動し，行く先々で現れる邪魔者と戦うシステムらしい。

昔，みんな楽しく生活しているところへいきなり雀魔王コクシーが現れた。魔王＝悪党で多くの人々が雀魔王に戦いを挑んだが，歯が立たずじまい。雀魔王はのさばるのであった！　ちょうど，ラジコンで遊んでいるＳのもとへＪが登場して，「お前のものは俺のもの，俺のものは俺のもの」の名文句を吐いた状況を想像すると，おわかりいただけるだろう(プライバシー保護のため実名は伏せさせていただきました)。

そんな状態が続くなか，雀魔王を倒すべく旅に出たテンパネル王国の聖雀士リードラー・リュウコ。彼女は放浪の末，フリテニス公国へと向かう途中であった。

「テクテクテク……むむ!?」

おっと，いきなり邪魔が入るとはご挨拶だぜ。お前のようなザコに用はないのだ！っとすかさず「逃げる」を選択。

「テクテクテク……ぬぬ!?」

また邪魔が入った。そうまでして戦いたいのならば相手をしてやってもよかろう！ふっ，なめてかかると火傷するぜお嬢ちゃん！……でもやっぱり「逃げる」を選択。何!?　回り込まれた？　くう，ここに進退極まったかー！

敵の魔物との戦いは当然麻雀である。和了った役によって相手に与えられるダメージの度合いが決まり，自分の体力が０になれば死んでしまう。こういうことを書くと必ず「えぇー!?　それじゃ体力が－１になったらどうなるんですかぁー？」と答えのわかり切った質問を投げかける輩がいるので，この際はっきりいわせていただく。マイナスになっても，し・ぬ・の・だ！

さて，いよいよ勝負。

そんな瞳で見つめても，手加減はしないからね

こいつぅ，嬉しそうな顔しやがって

ボク平和主義者だから，戦いたくないけどぉ

さあさあ，さっさと選びなさい

「パシ！　パシ！　パシ！　パシ！」
「リーチ！」
あれ？
「パシ！　パシ！　パシ！　パシ！」
「ツモ！」
あれれ？
「リーチ，ツモ，ドラ1」
「リュウコは力つきて死んでしまいました」
「GAME OVER」
…………え？
　あっという間の出来事だった。しかし，コンティニュー中毒の私はめげずにコンティニュー。だが，開始直後，再び惨事は訪れた。
「流局」「ノーテン」
「リュウコは力つきて死んでしまいました」

ツミコミ技はマジックポイントを消費する

「GAME OVER」
　リュウコ，なぜそんなに弱いのだ！　ユンケルでも飲んでもっとキバってくれ！
　しかたがないので，ツミコミ技を使ってみよう。ツミコミはマジックポイントを消費することによって行う。最初はマジックポイントが低いのでパイコウカンとイーペーコーぐらいしか使えない。ほかの手段として，剣を使ってツモる牌をよくしたり，盾を使って相手の手の内を一定時間のぞいたりすることもできる。剣と盾は品質がよいものほど効果バツグンだ。
　ふう，やっとフリテニス王国へ到着。ここでは女王陛下の娘リーシャンが雀魔王配下の者に連れ去られたらしい。雀魔王は身代金でもゆするつもりか？
　しかし，有無をいわさず娘救出を引き受けさせられてこちとら迷惑でい！　え？剣と盾とお金くれるの？　喜んでお引き受けいたしやすー。コクシー退治への道のりはまだまだ長く険しそうだ。
　ところでこのゲーム。町で人々が話しかけてくる場面や，建物に入ったときの「バフバフ……(私にはこう聞こえる)」というサウンドはうまく○ラ○ンクエストをパクっていて感動してしまった。そして，いま現在出回っている麻雀ゲームの多くがツモ＆捨て牌時間無制限なのだが，この「雀神クエスト」では時間制限がある。
「えーと，これを捨てるとこれが頭になってどれが待ち牌かな？　えーと，えーと」などと考えているうちに，無情のツモ切りを強いられる。いままでぬるま湯につかっていた雀プレイヤーは，したたる鼻水を拭う暇もないというわけだ。もちろんトイレに行きたいときにはポーズをかけられるが，その間はリュウコの入浴シーンに画面が切り替わる。ポーズをかけてじっくり考えようなんてズルを防ぐためだろう。

緑一色和了ってみせる!

　最近，コンピュータ麻雀やるなら2人打ちだなーと感じてきた。人間同士での4人打ちなら「昨日大家さんに滞納してた家賃総ナメされちゃってさー」などの会話で自分のツモまでの間をつなげることができるが，コンピュータ相手だと自分の番がくるまで非常にイラつくのだ。ただ単に私がセッカチなだけかもしれないが。
「PⅣ」は，バトルモードで2～3等身化された6人すべてのエンディングを見るのが楽しみ！
「麻雀航海記」はセーブ機能があるので，毎日，少しずつ遊ぶ時間を取りたい人にはおススメ！
「雀神クエスト」は暇をもてあましている若者にじっくり楽しんでいただきたい。BGMに関しては，「雀神クエスト」は制作側の気合を感じた。
　うーむ，「全部買ってプレイする」のがいちばんなんですけどね。

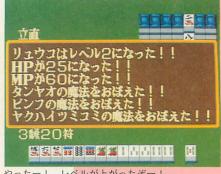
強い武器やお金が欲しいよー！

やったー！　レベルが上がったぞー！

マウスでぷちぷち打てます

　今回紹介したソフトは3作とも，マウスを使って打てます。ジョイパッドで牌を選択するのは親指が疲労でオーバーヒート気味になってしまいますし，キーボードでA～Nのキーで打つのは面倒。キーボード最上段の1～BSのキーもプレイ中に手を浮かせて打ち続けると手がだるくなって発狂してしまいそうです。やはり，マウス対応は正解といえますね。
　ただ，「雀神クエスト」は相手がツモ＆捨て牌を行っているあいだ，自分のマウスカーソルが消えてしまうのでちょっと不満でした。牌はみんな綺麗に表示されてました。

総合評価	0　5　10
麻雀航海記	
絵	★★★★★★★
音	★★★★
操作性	★★★★★★★★
雀神クエスト	
絵	★★★★★★★
音	★★★★★★★
操作性	★★★★★★

X68000ユーザーよゲーを磨け!

Nishikawa Zenji
西川 善司

X68000はソフトを作る人とそれで遊ぶ人の距離が最も近いパソコンであるといえるでしょう。商業目的ではなく，趣味のなかから生まれたゲームなどにもすぐれたものがたくさんあります。それらを収録した作品集の第２弾です。

MZ-5500アイドルは宍戸錠，パソピア７アイドルは横山父子，パソピアＩＱアイドルは岡田由希子，そして我らがX68000アイドルは山下章。というわけで，1991年７月から1992年４月にかけて全国規模で展開された，X68000アイドル山下章プロデュース/シャープ協賛のビッグイベント「第１回X68000芸術祭」。そして，そこでの優秀作品を収録したディスク付き書籍「The World of X68000」が，1993年末に電波新聞社より発売されている。

今回発売された「The World of X68000 II」はそのパート２ともいえるべきもので，優秀作品４作と前回収録された「FormulaX」の作者による新作ゲームを１作，合計で５作品を収録している。

それでは，１作ずつみていこう。

C力検査

小林康広氏作

テトリスにしろオセロにしろスイカ割りにしろ，長く遊ばれるゲームというものは，とっつきやすく，単純明快なルールのものが多い。プレイヤーは，ゲームに参加するために複雑な手順を踏む必要がなく，勝つための戦略思考だけに集中できるのだ。そんなわけで，昨日受精したばかりの受精卵から一昨日集中治療室にかつぎ込まれたお爺様まで，誰にでも遊べちゃう，と私が弱気に保証するのが，この「Ｃ力検査」ナリ。

さて，この「Ｃ力検査」はプレイヤーのＣ言語の知識を競うプログラミングクイズでもなく，ビタミンＣの摂取量を競い合うヘルシーアクションゲームでもない。じゃ視力検査？　惜しい，ちょっと違う。なんと「動体視力」と「反射神経」を磨く(競い合う)ゲームなのだ。いや自分の能力を育てる要素があるという意味合いから「EXERCISE」系という分類をしても国外撤去を命じられることはあるまい。運転免許の学科にこの「Ｃ力検査」を導入すれば交通事故は激減すると近所の魚屋は力説していた。関係諸官庁のみなさんご熟考あれ。

ルールは簡単。視力検査でお馴染みの「Ｃ」の文字，これが８種類のゲームモードに対応したさまざまな手法で表示される。この「Ｃ」の開いている方向をテンキー(もしくはジョイスティック)を使って正しく指し示すだけ。正しく入力できればポイント，できなければ１ミス(モードによってはゲームオーバー)。

では８種類のゲームモードとはどんなものがあるのか。

たとえば「サイズ変形」モードは，画面真ん中に毎回ランダムな大きさで，ランダムに４方向を指したＣが表示される。このＣが開いている方向を入力することになる。大きいＣのあとが突然小さいＣだったりすると一瞬入力を戸惑ってしまう。この「大きさ」のフェイントに惑わされるか否かがそのまま得点に結びつくわけだ。ステージが進むにつれて出題スピードが上がっていく。これだけの内容なのに一度始めたらなかなか止められない。スコア記録はモードごとに管理され，ベスト５はフロッピーディスクに記録される。他人とやれば熱い競い合い，ひとりならば自己ベストの更新への挑戦と，熱中指数は無限大だ。

そしてこのゲームには普通の２人同時プレイのほか，対戦モードがある。たとえ正しく正解しても先に答えたほうにしか得点が加算されなかったり，先に答えたほうが早く次の問題へ進めたりと，ライバルとの実力の差が具現化する。そして知らず知らず画面への集中度も高まる。コンマ数秒クラスのタッチの差で勝負が決まるのだ。ゲーム開始からゲームオーバーまで瞬きをするのを忘れて妙に目が痛くなっている自分を発見したり，横で発作を起こして倒れている友人を発見したり，そりゃもう大騒ぎさ。自宅で突然始まった宴会の盛り上げの道具としてもこの「Ｃ力検査」は使える。これがほんとの「宴会」ゲー，なんちてぼっくん。

C力検査
PLEASE SELECT!
サイズ固定
サイズ変形
ワンフレ
スピード
メモリー
16Cアタック
10Sアタック
トリプル

「Ｃ力検査」８種類もあるゲームモード

X68000用　5"2HD版　4,900円(税込)
電波新聞社　☎03(3445)6111

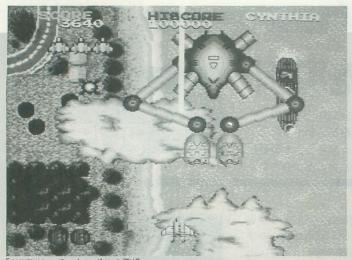
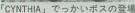
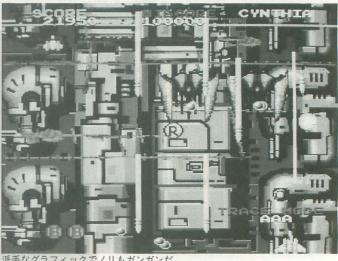
「16Cアタック」ではタイムを競う

突然の大きさの変化にドキドキしちゃうナリ

CYNTHIA

今橋晃一氏作

カレーを作るときにいろんなメーカーのルーを混ぜ入れてグルメの雰囲気を気取るような，東洋の影響を受けたギリシャ文明

「CYNTHIA」でっかいボスの登場

派手なグラフィックでノリもガンガンだ

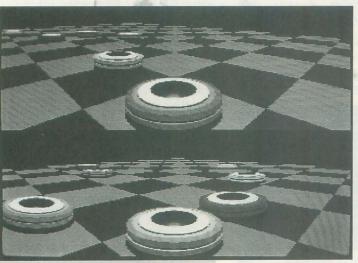
のような，目先の利益に束縛されない実に自由な発想と模倣の融合。やりたいことをやるアマチュアスピリットのよきお手本ともいえるのがこの「CYNTHIA」だ。とはいえ軽くみてもらっては困る。2HDのフロッピーディスク*1*枚がもつ容量を目一杯使って作られた，かなり本格的なシューティングゲームでもあるのだ。

ゲーム内容は，いわゆる強制縦スクロールタイプのシューティングゲーム。自機の動きは8方向レバーで操作し，Aボタンでショット，Bボタンで敵および敵弾一掃のボンバーという操作系である。3種類のアルファベットマークのボールを取るとショットがパワーアップ，同タイプのボールを重ねて取れば最大4段階のパワーアップができる。

BGMは，MIDIと内蔵音源のいずれかを選択できるというプロスペックな仕様。ゲームを起動するといきなり派手な音楽とともに文字タイトル，ボスキャラの画像が拡大縮小処理されながら画面奥へ吸い込まれていく。いまさら拡大縮小処理は珍しくないかもしれない。しかし，私の初代X68000/10MHz機で実現されているとは信じがたいほど高速であったため，驚きは隠せなかった。

ゲームを開始すると，エグザクトばりのラスター処理された雲がスクロールしてくる。ひき続いて画面狭しと回転処理バリバリの多関節カニ触手をもった中ボスの登場。なんだかすごいぞ，こりゃ。うわっ，敵の攻撃が当たる！　こうなりゃボンバーだ！ドカーン。ボンバーの爆発パターンも画面いっぱいに拡大処理されて表示。オオ。なんだかんだで，あっという間に1面のボス。背景の山に着陸していたと思われる1面ボス君は登場のテーマに合わせて自機と同じ高度まで急浮上してくる。ここでも拡大処理。ダハーン。

とにかく画像処理が派手だ。それも68000/10MHz機でも「高速」といえるくらいの速さ。うーん。こりゃすげーぜ。

確かに，グラフィックは粗削りだし，ステージ進行もありがちなとりとめのないものだけれど，難易度バランスは上々，単なる画像処理応用アプリケーションに終わっていない。ちゃんと「燃え」られる爽快感あるシューティングゲームに仕上がっている。

「RUSH!!」つきとばしゲームは対戦もできる

RUSH!!

京大マイコンクラブ/薄田昌広氏作

穴のあいた平面上で「おはじき(円盤)」を操作して，敵円盤とぶつけ合い，穴もしくは平面外にはじき出されたらアウト。視点は3D。簡単にいっちゃえばナムコの「モトス」の3D版。落ちるか落ちないかの崖っプチでのデッドヒートの爽快感と緊張感は本家をはるかに淩ぐ。

使用するのはレバーだけ，同じ方向にレバーを入れ続ければ加速し続け，逆方向に入れればブレーキになる。スピードが乗っているときに相手にぶつけてやれば，それだけ相手を遠くへはじき飛ばすことができる。フィールドには自分も含めて全部で8機の円盤がいる。自分以外は全部敵。勝者は1機のみだ。もし，自分が早くやられちゃっても退屈はしない。ゲームに関与できないけれど幽霊になってフィールド中を自由に動きまわることができ，他人のバトルを任意の視点で観戦できちゃうからね。

このゲームも対戦ができる。RS-232CでX68000同士をつなぐことによって，4人同時対戦。または，MIDIインタフェイスを使用してX68000を4台つないじゃえば，フィールド上の円盤を全部人間プレイヤーにし

今回唯一のツール「USEFUL」

て８人同時対戦バトルロイヤルが開催できる。う～ん，MIDIインタフェイスは確かに通信の一種だから，こういう使い方も可能なんだね。

ルールも簡単だから，起動した瞬間から遊べる。これが本当の「瞬間」ゲー，なんちて，ぽっくん(みすてないで……)。

USEFUL

新田隆仁氏作

「USEFUL」はスプライトエディタ。

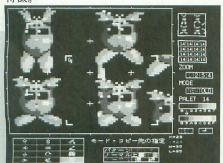
文書を書くツールには，手軽でスピーディな「エディタ」と高機能な「ワープロ」の２種類があるけれど，この「USEFUL」はどちらかといえば「エディタ」的な存在。必要最低限の機能を装備し，高速でかつスムーズな編集を目的としている。

「USEFUL」は画面を２つもっていて，１つがパレットの管理やパターン定義先を一望できるマネージメント的な画面，そしてもう１つは実際のパターンを作成する編集画面である。これらは，マウスカーソルを画面端にもっていくだけで瞬間的に切り替わる。マネージメントウィンドウでは作ったパターンのアニメーションテストもできるし，編集画面では１度単位のパターン回転機能もある。「必要最低限の機能」とはいっても，最近のたいていのニーズには対応しているのだ。

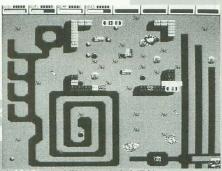
T-94X

Team ACQUIRE作

その独特のゲーム性と画面デザインで，それまでのカーレースゲームの常識を打ち破った「FormulaX」を制作したTeam ACQUIREが，今回この「THE WORLD OF X68000 II」のために新作ゲームを書き下ろした。その名も「T-94X」。

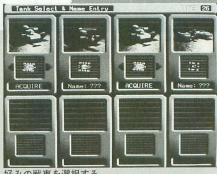
前作と同様，多人数同時参加型アクションゲーム。彼らの作風なのか，今回も512×512ドットでの高精細グラフィックが美しく，そしてスクロールなしの固定画面制が特徴。

軽さに味があるのだ

「T-94X」バトルロイヤルで生き残れ！

好みの戦車を選択する

攻撃だけでなく避けることも大切だ

いろいろなフィールドで楽しめる

プレイヤーは，性能の違う５種類の戦車(タンク)から１台を選択，これをレバーで操作，トリガで武器を発射する。敵の攻撃を受けるたびにダメージゲージが上昇し，ダメージゲージ100%でそのタンクは機能を停止する。壁や障害物を撃つと，武器パワーアップのアイテムが出現し，これを取るとタンクの性能が一定期間向上する。いかに敵から逃れるか，いかに攻撃を敵に命中させるか，これにバトルフィールドの地形特性と戦車の性能特性を絡めた戦略性が，このゲームの面白さだ。

ゲームは大きく分けて２種類。１つは「バトルロイヤル」モード。４～８台のタンクが一斉にフィールド場に放たれ，殺し合い，誰が生き残るかを競うものだ。ずばり，あのハドソンの「ボンバーマン」の戦車版といった感じだ。でも，「病気」アイテムはないよ，セニョール。

もう１つは「チームバトル」モード。これは２つのチームに別れ，フィールドに落ちているターゲットビットを拾い集めて，最終的にどちらがビットを多くもっているかを競うもの。せっかく取ったビットも，敵の砲撃を浴びるとその場に散らばってしまう。そこを狙いすましたようにチャッカリものがワサワサと集まってきて，それはオレんだよボケッ，とかケンカが始まったりして，和やかだったマイルームはいつのまにかサル山のディナータイムのように荒れ狂ってしまう(かもしれない)。

どちらのモードも，１～８人プレイが可能。１プレイは結構地味なので，やはり人間と対戦したいところ。夜，彼女とプレイしてて，わざと負けたりして，ハハハでもベッドの上では僕のタンクが勝つぞぉユカリちゃん，いやーんセイジさんたら～，ってやってたらコロス。

IIIも出るかな？

今回収録されたゲーム４作品はすべて，音楽制御にZ-MUSICを使用している。そのため，Z-MUSICを差し替えるだけでRS-232CやPOLYPHONを用いた(純正インタフェイス以外の)MIDI出力によるMIDI楽器BGM演奏が楽しめる。しかし，メーカーが保証しているわけではないので，これは各自の責任において試そうね。

さて，この「The World of X68000 II」は，書籍＋ディスク４枚という構成で，書籍のほうには開発者による開発エピソードが収録されている。苦労した点や開発環境などについても言及されているので，これから何かを作ろうとしている人には参考になるかもしれない。また，今回の作品のゲームのBGM楽譜も掲載されている。さらには，前回好評だった市販ゲームの「ウラ技・隠れ技」が今回の「II」でも掲載されている。ハマったままのあのゲームこのゲーム，裏モードでいきなりエンディング!! ……なんて邪道な楽しみ方もできるかも。

ところで，もし「III」があるんなら，今度はソースリストをつけたり，技術的な記事なども欲しいかな。

総合評価　0　5　10

熱中度	★★★★★★★★
お買い得度	★★★★★★★★
宴会ゲー	★★★★★★★★★

TREND ANALYSIS

1994年6月号のハガキ集計ベスト10 最近買って気に入ったソフトは？

POINT	タイトル	発売元	発売日
88	ジオグラフシール	エグザクト	'94/3/15
45	スーパーリアル麻雀PⅣ	ビング	'94/4/27
32	大魔界村	カプコン	'94/4/22
	ストリートファイターⅡダッシュ	カプコン	'93/11/26
29	ぷよぷよ	SPS	'94/3/25
27	アルゴスの戦士	電波新聞社	'94/4/27
23	餓狼伝説2	魔法株式会社	'93/12/23
20	悪魔城ドラキュラ	コナミ	'93/7/23
14	EasydrawSX-68K	シャープ	'93/8/5
10	あすか120% BURNING Fest.	ファミリーソフト	'94/4/22

（無作為抽出した1000通のハガキを集計）

エグザクトの「ジオグラフシール」が1位をキープしています。しかし，ポイント数はややダウンして2桁台に落ち着いてしまいました。ここのところ毎回，ポイント数の3桁レベルでトップ争いが行われていただけに，これはちょっと寂しい結果のような気もします。とはいえ，前回（5月号）の時点ではベスト10中，最新のソフトでしたが，今回は後発の5タイトルを抑え，しかも差をつけて余裕の1位。健闘していますね。何度でも繰り返し遊ぶタイプのゲームですから，やはり熱中度はとても高いようです。「毎朝，タイムアタックをやってから出かけるのが日課」なんていうコメントもありました。

さて，今月は初登場のソフトが10本中5本と半分を占めていますが，これは先月号で集計がお休みだったことも一因でしょう。また，4月後半に新作が次々と発売されたこともひとつの理由だと思われます。ゴールデンウィークにどこにも出かけないでゲームにどっぷりと浸かっていた人もいるのでは……？ ま，お気に入りのゲームに出合えたならば，幸せな休暇だったといえますよね。やはり人気の「スーパーリアル麻雀PⅣ」。難しすぎるのでは，と懸念されながらもしっかり上位に食い込んだ「大魔界村」。ナミダを流して喜んだ人も多いと思われます「アルゴスの戦士」。前作のあとのスピード開発でファンを驚かせた「あすか120% BURNING Fest.」。ラインナップはなかなかです。

ちょっと変わったところでは，9位に再浮上した「Easydraw SX-68K」。なんと1994年11月号以来のランキング入り。しかも14ポイントとなかなかの支持率。もっとも，ベスト10には登場こそしていませんでしたが，はがき自体は常に寄せられていたのですし，ゲームと違って流行に左右されないものであるだけに，今後もポイント増加の可能性もあります。そういえば，ベスト10の常連だった「MATIER ver2.0」が前回から姿を消してしまっていることも何か関係あるのでしょうか。使い込んだユーザーが今度は別のジャンルのソフトを使いこなし始めているのかもしれませんね。

それにしても，ゲーム以外のソフトは，期待度，満足度ともに非常に高いようなのですが，実際にランキングの上位に出てくることは少ないようです。ツール類などは，日常生活用品のような感覚で，とりたてて評価の俎上には上がってこないのでしょうか。しかし，期待のソフトのほうに挙げられている「SX-WINDOW ver.3.1」や「XDTP」なども発売がせまっていますので，こちらのランキングにも影響してくるかもしれません。さあ，今後の展開はどうなっていくのでしょうか？

COMPUTER MUSIC

▼

［特集］

入門コンピュータミュージック

▲

パソコンにひととおりのことができる音源がついていて，
それを操るソフトウェアがあれば，
そこにはコンピュータミュージックの扉が開いています。
X68000はそういった意味で比較的恵まれた環境にあったといっていいかもしれません。
それだけにかなり盛んに音楽データが作成されてきました。
音源を極めたデータはグラフィックと並んでX68000の特徴を際だたせています。
さらにMML式の音楽ドライバだけでも10種類以上発表されています。
それはコンピュータミュージックに希求するものの大きさを示しているといえるでしょう。
自分では楽器の演奏ができなくても，コンピュータを駆使すれば，
どんな難しい曲でも演奏できるようになります。
さらに手軽に，さらに多彩に，
一緒にコンピュータミュージックの世界を見つめてみましょう。

C O N T E N T S

基本はデータ入力から
脱初心者のコンピュータミュージック

Nakano Shuichi 中野 修一

COMPUTER MUSIC

コンピュータミュージックといっても本格的なことを始めるにはいろいろと揃えなければならないものがあります。現在でももっとも手軽に行われるのはMMLによるものです。ここではMML入力の注意などを見てみましょう。

「コンピュータミュージック」といっても，いきなりパソコンで作曲をしましょというわけではありません。

その多くは「あの曲を自分のパソコンで鳴らしたい」という単純な思いから始まっていくのでしょう。

X68000には音楽演奏機能が備えられていますし，X-BASICなり，Z-MUSICなり，その他のツールなりを使うこともできます。

あとはデータを入れるだけ。そこには実に単純な図式があります。MIDIでリアルタイム入力をするのもいいでしょう。楽譜入力もいいでしょう。しかし，Oh!Xで発表されるデータはMMLという形態をとっています。ここではMMLを中心にしたデータ入力について考えてみましょう。

＊　＊　＊

Oh!X(MZ)史上初のミュージックプログラムが掲載されたのはOh!MZ創刊2号目にして有田隆也氏の絵夢絶斗面白玉手箱でのことでした。

それはBASICリストで240行ほどのバッハのソナタでした。

当時まだミュージックプログラムというジャンルすら確立していない時期でしたので，紹介している有田氏もよくそれだけのデータを入力したものだと驚いています。「ちょっとデータが『頭痛が痒い』といったところでしょうが，作者の苦労を考えれば全快するでしょう」と結ばれていました。Oh!Xの基本的なスタンスが，早くも確立されているわけですが，同じ号には遙かに長いプログラムリストも掲載されていたわけですから(ちなみにTUX吉村氏のウォークワンだった)，ミュージックデータというものが一般に認知されていない状況，いい換えればコンピュータと音楽とがいかにかけ離れていたかがわかります。

当時は音楽機能といっても単音を鳴らすだけのもので，BASICに用意されたMMLを見ても音階と音量指定以外の細工などはなにもありません。

しかし，それでもMZというのは，当時もっとも音楽機能に優れたマシンだったのです。

現在では音源は楽器そのものですし，音楽ドライバやツールは完備されていますし，ミュージックデータもたくさん出回っています。はるかに環境が整っているといえるでしょう。しかしそれが入門者にとって恵まれているかどうかは微妙な問題かもしれません。

単にデータを入力するだけとはいっても，そこにもなかなかに深い世界があります。音源やドライバを駆使し，自分の思ったとおりの演奏を実現できるようになるまでには長い道のりがあります。

環境が揃った分，それだけ複雑になったというとらえ方をすると，それだけ便利になったはずなのに，難しいと思い込んでしまう人が結構いるということです。

こういった場合に，理解の障害となってしまうのは，環境が整っていないときなら誰でも自然に段階を追って身につけるようなことも，環境が整ってしまうと途端に難いものを目にしてしまうという現象のためです。変に体系だてると必要以上にものごとが難しくなってきます。これは学校でのお勉強を考えてみればよくわかりますね。

X68000発表当時はコマンドシェルを使いこなすことは当たり前だったのですが(初心者でさえも)，最近のように便利になってくるとコマンドシェルが難しいという人さえ現れてきています。便利な機能が多少増えたからといってほかはほとんど大差ないにもかかわらずです。

ひょっとすると初心者にコンピュータミュージックを説くなら「MUSIC SX-68Kで楽譜を入力してみましょう」とやるよりも，OPMのレジスタに値を叩き込んでいくところから始めたほうが実は親切なのかもしれません。

脱初心者のすすめ

このような，恵まれているゆえの罠を克服するには，まずは自分でデータを作成してみることです。マニュアルを読むだけでも必要十分な知識は得られますから，最初は独力でやってみることでしょう。こういう場合は曲のデキよりも完成させるということのほうが重要です。完成度自体はあとでいくらでも上げられるものです。

さらに上を目指すなら，さまざまなノウハウの部分が必要になります。そういったものの勉強法というのはデータ入力するところから始まるのでしょう。初心者のもっとも嫌いなプログラム入力です。およそ非人間的な行為を強要されているとして初心者の人権擁護を主張する声も多く聞かれます。

しかし，MMLの記述に慣れるにはいちばん手っとり早い方法であることも確かでしょう。単なるデータ配布の形態であればもっと違うかたちのものもありえます。いろいろ検討されているようですが，現状ではOh!X LIVEのリスト掲載のかたちをとるのはしかたありません。この場合，むしろデータ配布は二義的なものとして考えるべきでしょう。そこにあるのはノウハウなのですから。

最初は誰でも初心者です。しかし，いつまでも初心者えあることに甘えていてはいけません。ある人にいわせると初心者とは「なんの努力もしようとしない人」だそうです。逆にいえば，向上心を持っている人はもはや初心者ではないのです。

データ入力の方法

人によってやり方はいろいろあるでしょう。

Oh!X LIVEに掲載されているのはほとんどがZMSファイルですので，まずエディタを立ち上げます。なんにせよED.X(でなくてもいいが）の操作に習熟しておいて損はないでしょう。キーボードマクロの使い方に習熟するだけでもかなり作業効率は違ってきます。そのほか細々したものが効率を上げていきます。

たとえば，ED.X上から，

F1　先頭行に移動
CTRL+W　ファイル出力
OPM　ファイル名指定

といった操作で編集中のテキストファイルを演奏することができます。

ピュアの起動など，

ESC+C

による子プロセスの起動も多用されます。Z-MUSICならZ-MUSICシステムver.2.0に収録されていたZAM.Rが開発，デバッグに効果を発揮します。

テキストエディタ上でなんらかの作業をするというのはコンピュータでのあらゆる処理の基本になります。

ZMSの2つのスタイル

Z-MUSICの音楽データの場合，大きく分けて2種類のプログラムスタイルがあります。

ひとつはトラックごとにまとめてデータを記述してあるケース，もうひとつは先頭から順に全トラックまとめて記述してあるケースです。すなわち，トラックバッファの記述が，

(t1)
(t1)
(t1)
⋮

のような配列になっているか，

(t1)
(t2)
(t3)
⋮

のような配列になっているかということです。これらはデータ作成者によってたいていどちらかに統一されていますが，最近はトラックごとにまとめてあるデータが多くなったように思われます。

さて，トラックごとにまとめて記述してある場合は，メインメロディを担当しているパートから始めたほうがわかりやすいかもしれません（もっとも量が少ないトラックからやっていくという手もありますが……)。リストは，なにも先頭から順番に入力しなくてはいけないわけではありませんから。

また，同時に3つくらいのトラックをまとめて入力していったほうがわかりやすいという場合もあります。

同様に，先頭からまとめて記述してあるケースでも，すべてのトラックを入力するのではなく，トラックごとにデータを打ち込むというやり方もあります。複数の音源が組み合わせてある場合などは，音源ごとにばらして打ち込んだほうがいいかもしれません。

＊　＊　＊

どちらのアプローチにせよ，大事なのは，「すべてのデータを入力しなくても演奏を聞くことはできる」という事実です（もちろん途中まででですが)。ループの途中だった場合などはつじつまあわせが必要になりますが，データの最後に，

(P)

を入れておけば入力したところまでをいつでも再生できます。

もしもデータをすべて打ち込んで誤りがないことを完全に確認してからでないと演奏できないとしたらそれはあまりに不便すぎます。入力作業は単なる苦痛でしかないでしょう。

入力効率を上げるためにも，MMLの勉強のためにも，打ち込んだ部分のデータを聞き，確認しながら入力するというのはたいへん重要なことです。

データを確認する

打ち込まれたデータにはたいてい入力ミスがあるものです。パラメータ範囲や文法の整合をチェックするには，打ち込んだファイル名がTEST.ZMSならば，

ZMUSIC -C TEST

のようにZMDファイルにコンパイルすることによって，コンパイラが吐き出すエラーメッセージを参考にします。

なお，このときのメッセージはタグジャンプに利用できます。

さて，文法的に整合が取れていてもデータが間違っていたらなんにもなりませんので，それも確認する必要があります。音色番号や音程のズレは局所的なものですし，聞けばだいたいわかりますが（わからないものは直す必要がない？)，音長のズレは，それがデータ全体にかかわってくるため確実になくしておかなければなりません。しかし，ループなどが絡んでくるとどこでおかしくなったのか特定することが難しくなってきます。

そのようなときに役に立つのがカウンタ表示です。これだけで間違いが完全に解明されるわけではありませんが，少なくとも正確に入力できたことを確認することができます。

これは，打ち込んだファイル名がTEST.ZMSならば，

ZMUSIC-Q TEST

で表示されます。表示内容はトラックごとの音長値の合計が列記されたものです。トラック内にループがある場合は，ループ内の音長値の合計も表示されます。

どちらかといえばこれは積極的に間違いを探すものではなくて，完璧さを確認するという性格が強いので，このあたりは早急に改善が必要ですね。

そして……

手っとり早くノウハウを蓄積するには他人のリストを読むことが重要になります。音源に対する理解も必要になります。特にFM音源での音色作成あたりではノウハウと音源理解が猛烈に要求されてきます。Oh!X LIVEで発表されてきた曲の音色を片っ端から集めてみるとかもよいでしょう。

FM変調というおよそ感覚的ではない方式の音源ですが慣れてくれば勘所がわかって自在に音が作れるようになるといいますから，人間の潜在能力は計り知れません。

FM音源に比べれば最近のMIDI音源(SC-55など)はきわめてわかりやすい構造になっていますので音源の把握も比較的楽にできるでしょう。まあどこをいじっても突飛な音は出ませんから。

こうしてノウハウを蓄積ながら他人の作成したデータを聞くのもかまいませんが，やはり自分のお気にいりの曲は自分で作成してみるというのが本道です。いくらリスト入力に関して「脱初心者」してもあまり得るところはないでしょう。

別にこういったことはデータ作成に限りません。ツールやドライバなどでも同様なことがいえます。はっきりいって，音楽関係のツール作成は音楽に関心のない人にとっても興味深い題材がたくさんあります。

コンピュータミュージックには限りませんが，大事なのはいつまでも初心者でいてはいけないということです。「自らがデータの発信者になってこそ意味がある」とまではいいませんが，受け身ばかりでいるよりは作り手の側に立ったほうが充実した世界が広がっていくはずです。

COMPUTER MUSIC

愛と可能性を求めて

MUSIC SX-68Kを使う

Taki Yasushi 瀧 康史

待望のSX-WINDOW用楽譜ワープロの登場です。これさえあれば初心者でも簡単に曲データの作成ができるようになります。OPMDRV3のみならずZ-MUSICにも対応しています。

太平洋よりも大きな愛を心に持てる人はなかなかいないものです。ハイパーブラザーのような包容力のある愛。なにもかもを包み込む愛をいまのX680x0は必要としているのかもしれません。

私も日夜，その「愛」を育んでいるところですが，なかなか「愛」は育めないものです。ストレスからくる怒りがそれを邪魔するからです。やっぱりハイパーブラザーになるには，筋肉トレーニング以外に精神トレーニングが必要なようです。

さて，MUSIC SX-68Kです。

私が最近さんざんいっているとおり，これはSX-WINDOWのアプリケーションなので，あくまでもマルチタスクシステム上でデータを(ある程度)共有しながら作業を進めることができるはずです。操作体系も(ある程度)統一されているでしょうから，おそらく使いやすいはずです。

とまあ，ここまではSXユーザーならば誰でも予想でき，これにかかる期待はひとしおのものがあるでしょう。

特にMUSIC PRO-68Kを使わざるをえない立場にいた人ならば，もういうまでもありません。

MUSIC SX-68Kは基本的にMUSIC PRO-68KのSX-WINDOW版です。操作体系はSX化されたことによって一新しましたが，MUSIC PRO-68Kを使っていた人でSXユーザーならば，すぐにわかる操作体系を持っています。もともとのMUSIC PRO-68K自身，マニュアルレスでもけっこう使うことができるツールでしたから，MUSIC SX-68Kもこれまで楽譜入力ソフトを使ったことのない人でも，試行錯誤を繰り返すうちに使いこなすことができるようになると思われます。

このソフトの説明はMUSIC PRO-68KをバージョンアップしたSX版だといえば，話は済んでしまいます。しかし，MUSIC PRO-68Kそのものを使ったことのない人のために，概要を説明することにしましょう。

MUSIC SX-68K　38,000円(税別)
シャープ　☎03(3260)1161

楽譜ワープロ

このソフトは楽譜入力によるシーケンスができますが，その基本は楽譜ワープロというところにあります。MUSIC PRO-68Kも実は楽譜ワープロだったのですが，出力してもドットが粗く，画面のものをそのままハードコピーしたような感じで使いものになりませんでした。

MUSIC SX-68Kでは大いにその辺が見直されたようで，エディット画面は標準モード，縮小モードのいずれかを選ぶことができますし，印刷も縮小，標準，拡大の3パターンが選べ，出力はかなり綺麗になりました。

このサイズ設定は単なる同一データの，縮小，拡大ではなく，それぞれにつき，ひとつずつのパターンを持っているので，拡大してもギザギザになることはありません。

昔，私が音楽連載をもっていたとき，楽譜の掲載には非常に苦労した覚えがあります。その理由は自分の手で楽譜を書く方法がMUSIC PRO-68Kしかなかったからです。MUSIC PRO-68Kの印刷はとても誉められたものではありませんでしたが，MUSIC SX-68Kの出力のクオリティは，雑誌などの図で使用するに十分耐える出力です。

さらに記号なども増えています。文字ならば，半角単位で打ち込むことができるので，めったに使わないような音楽指示も書くことができます。全角が使えないのが難点ですが，それでもMUSIC PRO-68Kから比べれば大きな発展です。ついでに，パターンエディタのPAT4形式や，全角漢字，Easydrawのデータ形式も張り付けられれば完璧だったんですが。ちょっと残念ですね。特にEasydrawのデータ形式ならば印刷してもクオリティが劣化しないので最適だったのですが。

話は変わって，楽譜エディタと楽譜ワープロを共用すると起こる弊害として，楽譜に表示できないニュアンスと，シーケンスできない楽譜表示のギャップがあります。

このギャップをいかにして吸収できるかは今後楽譜入力ソフトを制作するうえでの大きなテーマになるでしょう。MUSIC SX-68KはMUSIC PRO-68Kからは随分前進しているものの，この辺は少しずぼらなようです。

MUSIC PRO-68Kからの発展として，装飾音（楽譜には記すが，拍があわないのでシーケンスデータとしては出力されない）の付記，スモールサイズの音符，歌詞や半角英数字の自由な張り付けができるようになりました。これだけ発展しているだけでも，よいとは思いますが，欲をいえばコントロール記号を楽譜上に埋め込まなくてはならないのをどうにかしてほしかったところです。

実際にあるデータなのですから，埋め込むのはしようがないですが，たとえば，MIDI番号，リズムのON/OFF，パートの指定のコントロール関係の表示は画面上には表示しても印刷はしないでほしかったところです（もちろん印刷するモードがあってもよい)。このあたりは次のバージョンアッ

基本の編集画面

さまざまなメニューが開く

プ時に期待したいところですね。

裏技でZ-MUSIC対応

裏技でZ-MUSICでの使用も可能です。SX-WINDOWの統一環境を乱しているのは，Oh!X編集部なので，あまりいえたほうではありませんが，これは非常にうれしい対応です。裏技というのは福袋のディレクトリに入っていて，メーカーサポート外の機能なので，ということです。若干，OPMDRV3.Xを常駐させたときとは動作が異なるようですが，使用する分にはまったく問題ありません。

ただインストールにコツがあって，失敗すると起動しないので注意しましょう。インストーラを立ち上げるときには，リセットしてもかまわない状態で立ち上げるでしょうから，それほど問題ないでしょうけれど，福袋に入っているReadMe.DOCをよく読んでから実行したほうがいいですよ。

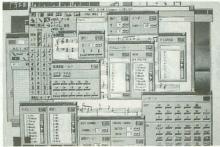
当然ですが，これはイリーガルな使い方なので，残念ながらSOUND SX-68Kと二人三脚で使うことはできません。OPMDRV3.Xならば，一緒に使えます。

どちらを常駐していても，ZMSファイルやOPMのいずれも出力できるので，ほかになにも使わない人はOPMDRV3を常駐していたほうがよいかもしれません。

私としては，Z-MUSICとか，OPMDRV3よりも，サウンドマネージャを作ってくれたほうがSXらしくてよいと思うのですが……。

ちょっとつらいところ

でもちょっとだけつらいところがあります。まず最初につらいと思ったのはビームの指定です。ビームというのは音符の髭と髭をつなぐアレなのですが，MUSIC PRO-68Kのときよりは進化したものの，結局同じようなことをしなくてはならなくなってしまったことです。

SXでマウスらくらくなのですから，ビームもマークしてオートでやってくれればよさそうなものです。確かにやってくれることはやってくれるのですが，斜めにならず，水平になってしまうのです。斜めにするためにはMUSIC PRO-68Kと同じ手順がいるので，結局あまり変わっていないということです(SXなので軽い分，少しは楽になったけど)。

不満な点はほかにもあります。

タイやスラーなのですが，長さが一定のものしかないということです。実はドローイングスラーというものがあって，なんだかよくわからない計算アルゴリズムによって長かったり短かったりするスラーをひけるのですが，このなんだかわからない曲線計算アルゴリズムが曲者なのです。どう変なのかよくわからないのですが，とにかくなんか変ですし，ドローイングスラーの部分だけ印刷が汚くなってしまうので結局使っていません。

そのほか，FM音源部の8音，ADPCMの1音はいいとしてもMIDIで合計16音しか出せないのはちょっと変です。

あと，変な仕様がいくつかありまして，ブロックの最終小節でビーム指定をすると，その辺一帯が消えてしまったりします。いけにえとして，あらかじめ隅っこに適当な音符を置いておけば回避できるので，実害はないのですが，そのほかのところが割とよくできているだけに，残念なところです。

まとめ

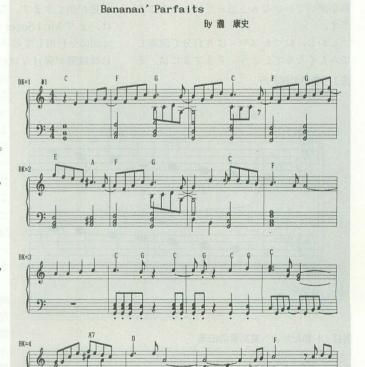
「買い」かどうかは読者の皆さんが決めることですが，私の独断と偏見から判断するならば，楽譜を印刷したい，Z-MUSICでデータを作ってみたかったという人は買いでしょう。

先ほどいったとおり，OPMDRV3.X版のMUSIC SX-68Kでも，作ったデータのZMSファイルは書き出せるので，SOUND SX-68Kとの統合環境でZ-MUSIC用のデータを作ることもできます（いいのができたらOh!Xに投稿してね）。これは面白いですよね。

そもそも，このソフトは楽譜印刷が綺麗になっただけでも存在価値があるわけですし。これで楽譜入りの文章が書きやすくなったかな？

SXのアプリもどんどん揃ってきました。たまには大きな愛をもって制覇せねばならないソフトもあるようですが，短気にならずに使えば，かなり面白い使い方ができるものです。

割と人間って勝手ですから，思いどおりに使えないといらついちゃうものなんですけどね。

図　出力例

楽典の基礎知識
楽譜が読めるようになるために

Taki Yasushi 瀧 康史

COMPUTER MUSIC

期待の楽譜入力ソフトMUSIC SX-68Kも登場し，ミュージックデータの打ち込みもずいぶんとっつきやすくなってきました。ここでは市販の楽譜を読むための基礎知識をまとめてみました。

ちょっと奥まで音楽してみよう!

待ちに待った音楽の特集のお蔭で，久しぶりに，この雑誌で音楽の仕事をすることになりました。

さて，数カ月前に終わった私の音楽連載では，最初から楽譜は読めるものとして始めました。楽譜が読めなければ音楽しちゃいけないかっていうとそんなこと全然ないですよね？　音楽するって，音楽を作曲するだけじゃなくて，演奏したり，聞いたりすることも音楽することだと思います。

なにも知識はなくても，音楽を聞いて，「いいなぁ」と思うことはできます。人間の感性は自由で無限ですから，たくさん音楽を聞いていいなぁと思ってほしいところです。

しかし，いつかはやっぱり自分で演奏してみたくなるでしょう。そうするには，まず楽譜を読むことが必要ですし，本来ならば楽器の練習も必要になります。

楽譜が読めるようになることは割と簡単です。一夜漬でも簡単に身についてしまいます。しかし，楽器の練習は一朝一夕にできるものではありません。日々の鍛錬が必要になりますよね。それと，練習のための時間も当然必要となってきます。

そこで，DTMというものが出てきます。そうです。デスクトップミュージック。多分ローランドの造語だと思うんですが，まぁ語源なんかよりも内容。要は机上音楽。演奏はコンピュータにお任せ。人間は演奏の手順を教えてあげるだけ。

となると，人間は楽譜が読め，楽譜をなんらかの形でコンピュータに教えてあげる必要が出てきます。なんらかの形というのは，まずMu1Super（Mu-1GS），Music studioを利用してのリアルタイム録音。これは鍵盤が弾けなければ話になりません。

図1　音符の読み方

ト音記号
c d e f g a b c̄
この2つは同じ
へ音記号
c̲ d̲ e̲ f̲ g̲ a̲ b̲ c

次はZ-MUSICのようなMMLによるデータ入力。これには，MMLという記号を覚えなくてはいけません。ちょっと一朝一夕にできる代物ではないですよね。この仕事も随分長いのに，いまだにまともにMMLで音楽が書けない組めない人がここにもいますし。

そこでMUSIC SX-68Kに注目です。残念ながら，(私としては)期待通りのデキではありませんでしたが，楽譜の読み方さえわかれば，簡単，お手軽に音楽できるこの極み。

いままで音楽するのも聞くだけだったあなた，さらに一歩踏み込んで，DTMの世界まで，ちょっと奥まで音楽してみましょう。

図2　ト音記号とヘ変記号の由来

音符の読み方と調号

おたまじゃくしは蛙の子？　なんて，この手の説明には必ずといってよいほど出てくる比喩ですね。そうです。「おたまじゃくし」とは音符のこと。

図1を見てください。

これは，ト音記号とヘ音記号での音符の読み方を記してある図です。ほとんどの人はこの辺りのことはわかると思いますが，音符以外のいちばん左の記号で，上の記号はト音記号，下の記号はヘ音記号といいます。

MMLで表記した場合，ト音記号の最初のcは"o4c"と表し，ヘ音記号の最初のcは"o3c"と表します。図中ではcdefgabと書いてありますが，これは音符の読み方です。日本では，「ドレミファソラシド」というのが，メジャーな読み方ですが，海の向こうでは「cdefgabc」という読み方をするということです。

ヘ音記号とト音記号は，1オクターブ違うのですから，ヘ音記号の欄のいちばん右の音符と，ト音記号の欄のいちばん左の音符はまったく同じ高さの音です。

参考までにどうしてト音記号とヘ音記号というのか図2に載せておきます。ト音記号はGを模した記号の中心がG，すなわちト音であること，ヘ音記号はFを模した記号の中心がF，すなわちヘ音であることからきています。

次にいきましょう。

図3は，音符の長さについての説明図です。まず，最初にある4/4という記号は，分母が4分音符を意味し，分子は数を意味します。つまりこれは，「4分音符が1小節に4つある」ことを意味します。

小節とは，線で囲まれた内部のことをいいます。これは楽譜をうまいタイミングで区切っている線のことです。この小節線で囲まれた内部がつまり，1小節になるわけです。

さて，まず最初の小節には，全音符が書いてあります。これは，小節すべてを埋めてしまう音符です(4/4拍子時)。Z-MUSICのMMLでは，この音符はラ，つまりaですから，o4a1と表記します。次の小節は，2分音符です。この2という数は，2小節に2つまで入るということを意味します。つまり長さは全音符の1/2なわけです。図中のラとシ，つまりaとbでは棒の向きが違います。これはこのラとシの間で棒が入れ替わると決まっているからです。

2分音符は1小節に2つまでしか入れることができませんから当然2つしかありません。4分音符は，最初の4/4の記号通り，これは4つあるはずです。このようにして，何分音符というのは1小節にいくつ入れられるかによって決まります。基本的に髭の数が増えていくほど音の長さがどんどん短くなっていきます。Z-MUSICのMMLでは，4分音符ならa4，8分音符ならばa8と表記します。音程の次の数字は音符の長さを意味しているのです。

図4は休符です。休符とは，音が鳴らない，つまり休めの記号です。書き方にもやはり法則性があって，8分休符からは髭が徐々に増えるだけになります。

図5はさまざまな音符の長さについての説明です。1小節目です。付点がつくと，その音長の長さの半分を足すことを意味しています。1小節の残りが4分音符であることから明白でしょう。さらに2小節目は，これで1小節目と同じです。この横の長い弧のようなものはタイといいまして，前後の音をつなげるという役目を持っています。

3小節目は，休符にも同じように付点が使えることを意味しています。

4小節目にある3というのは，3連符といって，括弧で囲まれた部分を等分して鳴らすという記号です。長さから考えればわかるとおり，最初の3連符は2分音符の長さを3つに分けて鳴らすという指示，次の3連符は4分音符の長さを3つに分けて鳴らせという指示になります。1番目と2番目の3が上と下にあるのは，どちらでもかまわないという意味です。

音符，休符の長さに関する説明が終わったところで，もう一度音の高さに対する説明をしましょう。

図6には図1と違って，なにやら，妙な記号がついています。#♭♮です。

それぞれ，シャープ，フラット，ナチュラルというのですが，シャープは音を半音上げ，フラットは音を半音下げ，ナチュラルはそれらによって半音上げたり下げたりされた音の高さを，元の高さに戻すという効果があります。音符の手前にこの記号がついていた場合，効果が持続されるのは，その小節の終わりまでです。つまり5小節目の最初のシ（b）には，ナチュラルがついていないことに注目。前の小節でフラットがついていますが，小節線をまたがったことにより，これらがクリアされたのです。楽譜によってはこのような場合，いちいちナチュラルを書く場合がありますが，これはわかりやすく書いてあるのであって，書かなくても間違いではありません。

図中の説明にあるとおり，C#とD♭は名前が異なった同じ音です。また，ミ(e)とファ(f)，シ(b)とド(c)の間には半音がないことに着目してください。MML表記では，a#4や，a-4のように，音符の直後にシャープやマイナス（フラット）を入れることによって再現できます。なお，シャープは＋で

図3

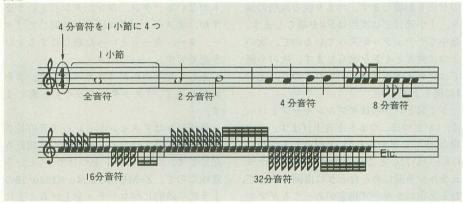



図4

図5

図6

もかまいません。

もしも，楽譜上やMML上でミやシに#がついていた場合，これは上の音のファ，ドを意味します。同様に，ファやドにフラットがついていた場合はミやシを意味します。

これらのような記号を臨時調号といいます。臨時調号はそのあとの音符からその小節の終わりまで影響しますが，臨時でない調号の場合，あえて解除されない場合，楽譜全般を通して影響します。その説明が図7です。

図7ではシャープがひとつFについています。上のFについていますが，Fなら下のFにも影響します。つまり図の説明の通り，1小節目では実際はF#を鳴らします。途中でナチュラルが入っているので，次の2つは両方ともFを鳴らします。次の小節ではナチュラルはクリアされるので，元の#に戻ります。

さて𝄪というのはダブルシャープといって，#が2つ分，つまり全音上げます。この場合，すでにFからF#に上がってますからG#といいたいところですが，これだとナチュラルが手前にあったときに混同するので，たとえ前にシャープ指定があってもダブルシャープは全音しか上がりません。また，下のAはダブルフラットがついているのでGとわかるでしょう。

演奏を多彩にする記号

音楽はときに小さく，ときに大きく奏でてメリハリと表情をつけていきます。その音の強弱を表す記号は，下から順に，*pp*(ピアニシモ)，*p*(ピアノ)，*mp*(メゾピアノ)，*mf*(メゾフォルテ)，*f*(フォルテ)，*ff*(フォルティッシモ)などと続きます。Z-MUSICでは@uコマンドがこれに相当します。@uコマンドは0から127までで音の強弱を表しますが，*pp*〜*ff*などの記号は人間の感ずるままの大きさなので，なにもいくつが確定というわけではありません。ともあれコンピュータでDTMするには，音の強さは決めざるをえませんから，自分で各自適当に決める必要性が出てきます。

音の長さにも音符では表せない長さを導入することで，その感情を高めることがあります。それらを楽譜に表しているのが，図8です。

まず最初はフェルマータです。これは，音をしばし長くとどめるという意味です。長さは演奏者の考え方次第です。Z-MUSICなどのシーケンサの場合，この音符だけテンポを半分ぐらいから3/4ぐらいに延ばすのがよいでしょう。

次はテヌートで時間一杯まで音符を延ばすという意味です。リズム的には，テヌートがないと「タン，タン，タン」といきますが，テヌートがあれば長さは同じで「ター，ター，ター」といった感じにするという記号です。

そしてただの点がスタカートです。音を歯切れよく，「タッタッタッ！」と奏でます。

最後の>はアクセントといって音の長さに関わる記号ではなく，音の強さに関わる記号です。「アクセントをつけて！」という意味なので，Z-MUSICではu+10au-10のように一時的にベロシティを上げることにより再現できます。頻繁にアクセントをつけたり，きめ細かい音量変化をつけたい場合はこのように記述していたのでは効率的ではない場合もあります。Z-MUSIC ver.2.0では発音指定が拡張されており，従来の，

c4　　(Cの音を4分音符で鳴らす)

といった音程+音長以外に，

c4,2,125

といった指定が可能になっています。これはステップタイム4(4分音符分)，ゲートタイム2で音量125の音を鳴らすという意味になります。ここで使われる音量は臨時ベロシティと呼ばれるもので，その音だけに有効な値です。ちなみにステップタイム4でゲートタイム2の音を鳴らすというのは，2分音符のCの音を鳴らしながら4分音符分の時間がきたら次の処理を開始するという意味です(現バージョンでは複数音には対応できない)。

さらに図9です。

まずこの図には，4/4という記号がありません。なにもないときは4/4ですが，これは単なる省略なので，自分で書くときはわかりやすくするべきですね。

そして，タイのように弧を描いているものがあります。同じ音符同士をつないではいません。これがスラーです。スラーはMUSIC SX-68Kには妥当な動作をするものがありませんが(一応ドローイングスラーというものもあるが……)，これらの音を滑らかにつなぐという意味があります。Z-MUSICではe8&f8&g4&g8としがちですが，FM音源などでは単にアタックを鳴らさずという意味になり，MIDI音源ではノートオフしないという意味にすぎません。弦楽器や金管楽器ではそれらしく聞こえないこともないですが，ピアノなどではあまり妥当なスラーとはいえないでしょう。

むしろこの間のゲートタイムQを可変して，Q8e8f8Q7g4&g8というように変えたほうがよろしいかと思います。これら，コマンドの深い説明はZ-MUSICのマニュアルを個別に参照してください。

注意すべき点はほかにもあります。まずこのソが，付点4分音符ではないことです。わざわざ4分音符と8分音符でつないだのは，まずこの曲が4/4の曲だからです。

音楽には拍というものがあって，この曲では分母の4から4分音符を拍にしています。拍というのは，音楽を聴くときに手を音楽にあわせて叩いたりする「あれ」のことです。つまり，拍をまたいで延ばす音は，

図8

図7

図9

楽譜上の記載ではタイなどでつないだほうが，奏者に見やすいということなのです。DTMをする場合，コンピュータは間違えませんから，別によいといえばよいのですが，綺麗な楽譜を書くという習慣はつけておくと便利かもしれませんので一応。

2小節目ではなにやら変な音符がありますが，単なる付点8分音符と16分音符の結合ですから，あまり気にする必要はありません。16分音符は次の8分音符とタイで結ばれているので，結果的にこれは付点8分音符，付点8分音符，8分音符という演奏になります。MMLで表記したなら，普通にg8.a16&a8g8といった感じになります。

最後の小さな音符は装飾音符です。MUSIC SX-68Kでは印刷専用記号なので無視されてしまいますが，本当は拍に入らないように音を鳴らします。数えたらわかるように，この小さな装飾音は，全体の音符の長さから除外して考えられています。

わずかにして，気持ち，鳴らせばよいのでMMLではg32a16.g8というふうにすればそれらしく聞こえます。

ループ記号

ループ記号は本来，楽譜を単純化するために作られたものなのでしょうが，例によって例の如く，あまりに使用しすぎると，どんどん見づらくなってきます。

奏者のためには，適度にしかループ記号を使わず，複雑になるのならばループ展開したほうがよいのでしょうが，楽譜を出版するに当たっての誌面の都合などから，実際に発売されている楽譜には，ループ記号は大いに駆使されています。

とりあえず図10によく使われるループ記号をまとめてみました。

まず，1の:‖はループ記号の代表的なもので，前の‖:に戻るという意味を持っています。最初のAには，‖:がないのでいちばん最初に戻ります。したがって，Dの終わりの:‖は2の‖:に戻ることになります。

Eのブロックでは，1小節が‖:と:‖に囲まれています。このままではC，Dと同じく1回繰り返すだけなのですが，3に3 timesという表記があるので，実際には3周回ります。

さて，お次は4の記号です。

これは1度目，2度目，3度目と途中で演奏順序を変えるという表記で，結果的にはGから，G，H，G，I，G，J，Kといった順に演奏します。

そしてKの下の5，D.C.に注目です。これはダカーポといって，いちばん最初に戻るという記号です。つまりAまで戻るということですね。再び同じように演奏し，Kまできたら次はあえて7を無視し，Lにいきます。そのまま8も無視し，Pまでいくと，10の記号，つまり‖は終端記号なのですが，その下に6，D.S.(ダルセーニョ)があるため，7のSのような記号まで戻ることになります。

再び，Nまでいったときは，今度は8を無視できません。8のtoCodaという記号は，1度目は無視しても，2度目は有効という特質があって，この記号は名のとおり，Codaにジャンプします。そして，終端記号‖で終わりとなります。

最終的な順序を追ってみましょう。

AABCDCDEEEFGHGIGJK AABCDCDEEEFGHGIGJKL MNOPLMNQRST。

暗号みたいでややこしいですが，ここまで複雑に楽譜が書かれることはまずないので，安心してください。

まとめ

寝不足で眠いことを除けば，久しぶりの音楽関連記事はなかなか楽しく書けました。面倒なハードウェアを作ってるよりも，遙かに楽だったりして。性にあってるのかもしれません……。

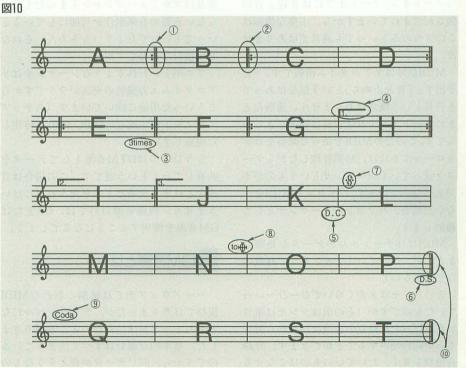
それはさておき，これで楽譜の読み方の基礎の基礎を記したつもりですが，あんまり資料などを見ずに書き始めたので，漏れがいくつかあることでしょう。

音楽記号はまだまだたくさんあって，私にも実は覚え切れないくらいあります。そういうのに出合ったときは興味があったら音楽標語事典で引いてみてほしいですね。ただ，だいたい今回説明しただけ知っていれば，まずたいていの楽譜は読めるでしょう。

MUSIC SX-68Kはいわゆる楽譜ワープロですが，簡単なシーケンサとして使うこともできます。ほとんど楽譜をそのままコピーするように打ち込めば，ある程度までDTMできますが，表現力に限界がありますので，凝りに凝ったデータにするにはZMSファイルで吐き出してMMLを加えていくか，SCOファイルからMu-1GSなどを利用してステップエディットするか，どちらかにしたほうがよいでしょう。

これ以上の知識は，私が昔連載していたCreative Computer Music入門の最初のほうに続きます。興味があって，向上心がある方は読んでみてください(もっとも，いま見ると結構恥ずかしいことをしていますが)。

こういう記事は音楽を知り抜いている人にはなんにも役に立たないものかもしれませんが，誰だって初心者から始めるものです。初心者の人もまだまだ悲観しなくてもきっと追いつけますので，頑張ってみてください。

図10

より高度な音楽環境のために

MIDIセットアップマニュアル

Nakano Shunichi 中野 修一

COMPUTER MUSIC

多少投資は必要ですが，データ作成の容易さからいうとMIDI機器は内蔵音源に勝ります。最近の音源は侮れません。その気になればかなり本格的なことができます。ここではMIDI導入の手引きを見てみましょう。

パソコンユーザーにとってMIDIも特別なものではなくなってきました。いまどきパソコンで音楽をやるといえば，もはやMIDIをつないでSC-55（相当品）で音を鳴らすことだといってもいいでしょう。

MIDI機器というのは，コマンドを送れば音源がそれを解釈して動作するという単純なものです。どのような動作指定が可能かというと，大まかにいって，

ノートオン
ノートオフ
プログラムチェンジ
コントロールチェンジ

となります。

ノートオンは音を出すこと，ノートオフは音を止めること，プログラムチェンジは音色を切り換えること，コントロールチェンジは演奏表現を指定するもの，といった対応になっています。

ノートオン/ノートオフには音程と音の強さが含まれていますから，主要な情報はこの2つのみといっても過言ではありません。

MIDI信号はリアルタイム情報です。「音を出す」「音を止める」という信号はあっても音長という概念はありません。適当なときにノートオフすれば音長は管理できるからです。このためMIDIを使って楽器をコントロールするには，時間管理しながらデータを送っていくシーケンサというものが不可欠になっています。パソコンにMIDIをつないだ場合，パソコンがシーケンサとして機能します。

MIDIは16チャンネルのデータを処理できますが，たいてい1チャンネルで複数の音を出せるので

昔のシンセは8音くらいでひーひーいっていたものですが（その前はシンセは単音が常識だった），最近の音源は24音から32音くらい同時発声することができます。登録音色数も多く，たいていのものはこなせるような仕様のものがDTMの主流になっています。

MUSIC SX-68Kあたりを使えばデータを作ることも簡単です。音源やシーケンサは高性能になってきた分ユーザーの負担は減ってきましたし，データの入手も簡単になりました。とりあえず音楽を始めてみるにはいまがよい時期といえるかもしれません。

MIDI楽器への2つのアプローチ

MIDIを導入してコンピュータで音楽をするには2つのアプローチがあります。

ひとつは音楽制作に特化したかたちをとるものです。この場合，MIDI機器などは自由な編成にすることができます。自分でテープ起こしなどを行えばプライベートにCDを焼くこともできますし，特にDATの普及はプロミュージシャンとまったく変わらない音質の音楽制作を可能にしているといってもいいでしょう（もちろん，それなりの機材は必要ですが）。

プロ用といわれるようなシーケンサはリアルタイム入力機能の充実がウリですから，こういった用途に向いています（ステップエディットが主流なのはPC-9801に特化した現象です）。

もうひとつはDTM音源としてデータを共有していくという道です。この場合は音源などはそう突飛なものを使うわけにはいきません。内蔵音源以外では，GSまたはGM音源を使用することになるでしょう。

GM音源とは

ハードウェア面では統制の取れたMIDI規格ではありましたが，データレベルになるとちょっとおぼつかないところがありました。基本的な部分はすべて決まっているのですから，同じデータが使えそうなものなのですが，これがなかなか難しい問題だったのです。最大の欠陥は楽器の音色番号を特定していないということでした。同じメーカーのものでさえ違う配列になっていましたので互換性は取りようもありません。

このような状況になった原因は，MIDI規格制定当時の主流はシンセサイザだったからでしょう。現在主流のものはプリセットサンプラです。要するに，用意された音色を使って曲を演奏するか，音色を作成するところから始まるかという違いがあります。

昔は，MIDIはプロの音楽制作に使われるもので，複数の音源を使うことは当たり前でしたし，音源を目一杯使って音楽制作をしていればどこにも同じ環境というのはなくなってきます。プリセット音色は用意されていてもユーザーにエディットされてしまうことは間違いありませんでしたから，同じデータで同じ音を鳴らすという発想がなかったのでしょう。せめて標準配列だけでも指定してあればずいぶん違ったはずなのですが。

一方，DTMという観点からいけば互換性という問題は非常に重要になってきます。そこで現在では音色をほとんどエディットできないプリセットサンプラが中心になっています。

各社バラバラだった音源の仕様を統一して，同じ音楽データを鳴らせるように出てきたのがGM規格です。

具体的に統一されたのは基本音色配列，発音応力（最低24声），サポートされるべきコントロールチェンジなどです。

現在のところ，事実上SC-55仕様で統一されています。GM用として発売されている音楽データの大半はSC-55で作成されたものですので，SC-55で再生するときにもっともバランスよく演奏されます。

初期のGM音源では各社の個性があった（ありすぎた？）のですが，これでは同じデータを鳴らしたときにバランスが非常に

悪くなるので，最近のGM音源は音の作り方がSC-55に近くなっています。なかにはSC-55をエミュレートするような音源も現れてきました。

もともとGM音源として音色の配列を決めても具体的なオクターブ指定や音量バランスなどが指定されなかったので正確なデータ演奏は望むべくもなかったのですが，先行したSC-55がデファクトスタンダードとして認知されてきたということです。

具体的な製品としては，ローランドSC-55(初代)，ローランドSC-88，ヤマハTG-300，コルグAG-10，EMU SOUND ENGINE(WAVE BLASTER)などが標準的といえるGM音源です。

GS規格とSC-88

GSというのはGM規格に対して上位となるようなローランド独自の音源規格です。GM音源がSC-55仕様に近づいているので大差ないように思われるかもしれませんが，スペックではGMの上をいきますし，互換性の取り方もはるかに厳密です（それでもまだまだ甘いのですが……）。一般に音質云々よりも「忠実再生」ゆえにGSはGMよりも高く評価されています。

GSというと，今月は紹介が間にあいませんでしたが，新製品ローランドSC-88に話題が集中します。製品としては従来あった２系統のGS規格，すなわちキーボードつきのJV系の音源と音源モジュールのSC系の音源を組み合わせた感じです。SC系とJV系のモードを持ちJV系のセッティングでは比較的上品な音色設定になっているようです。

ちょっと触った感じでは，SC-88はSC-55mkIIよりもSC-55（初代）に近いのではないかと思われます。SC-55mkIIとSC-55の互換性がいまひとつだっただけに，次期主力製品としての座は揺るぎないものになるでしょう。

GS規格の変更によりどうなるかと思われたデータ互換性ですが，これで結果的にSC-55mkIIが泣くという意外な展開になってしまいそうです。

DTMを支える機器たち

MIDIというと音源ばかりに目がいきがちになりますが，楽器以外にも重要な機器があります。もっとも一般的なものとしてはミキサーが挙げられます。なにをするかというと単にいろいろな機器からの音を混ぜ合わせるだけなのですが，本格的にDTMするなら必需品といってもいいでしょう。

普通は内蔵音源からの出力を混ぜるので２系統（4ch）は最低限必要です。気合を入れてDTMするなら鍵盤つきの楽器を加えることも考慮すべきですし，将来的な拡張というし，将来的にX68000に音源ボードが出てくるかもしれません。少しは余裕を持って８ch～12chくらいのものを選ぶのがよいでしょう。ゲーム機の音をエフェクタを通してスピーカに入れるとかいった用途も当然考えられますし，各自の予算と接地スペース，将来展望にあわせて選択してください。

一般的なミキサーというのは，

入力

　各音源からミキサーへ

エフェクトセンド

　ミキサーからエフェクタへ

エフェクトリターン

　エフェクタからミキサーへ

出力

　ミキサーからアンプ/スピーカへ

のような構成になっており，単に各種入力のバランスをとるだけでなく，各種入力ごとにどの程度エフェクタに送るか，エフェクタから返ってきた信号と源信号をどのような割合で混ぜるかなどが指定できます。通常はエフェクタに送るのは内蔵音源からの出力だけでいいでしょう。

ちなみにちゃんとつなぐと大量の音声ケーブルが必要になります（編集室の機材ではミキサーに入っているだけで17本）。

大きくなるとイコライジング機能がついているものが多いのですが，ミキサーというのはシンプルな構成がもっとも望ましい機器ですのでできるだけ単純なものを選ぶのが賢明です。

●エフェクタ

エフェクタにはだいたい単機能型，複機能型，多機能型，超多機能型といった４つぐらいのグレードがあります。

単機能型は面白みに欠けます。多機能なデジタルエフェクタは触っていて楽しいのですが，多機能な部分はあまり使えません。最近の高級な超多機能エフェクタは音に変化をつけるといったレベルではなく，入力信号を素材としてあたかも楽器のように振る舞います。こういったものを駆使した音楽制作というのもDTMのひとつの姿ですが，はっきりいってプロでも必要としないような機材に分類されます。

ということで，単純な処理をクリアにこなすものが優秀とされます。海外産のMIDIverbなどの人気が高いようです。

外部エフェクタは内蔵音源の出力にかけるべきものですが，機器の性格上，音を壊すものですので，多用は控えねばなりません。たいていの音楽データは生出力にあわせて調整されており，必要なエフェクトに相当するものは音楽データ中に盛り込まれていることも少なくありません。ですから過剰なエフェクトを加えると音が濁り，ドラムがベコベコになります。

エフェクタのかかったMIDIの出力の出力と親和しやすいようにごく弱めにかけるのがよいでしょう。FM音源から出るような硬い音はちょっとしたエフェクトを加えるだけで音の性格が変わることもありますので注意が必要です。

そのほか，BBEのSONIC MAXMIZER（入力信号波形の歪みをなくす機器）などを好んで使う人もいますが，たぶん普通の人が聞いても効果はわからないでしょう。もともとエフェクタに入れる前の信号を整形して音の濁りをなくすためのものなのですが，音源モジュールからの出力にはすでにエフェクトがかかっていることが多いのでDTMにはむしろ不向きな機器といえるかもしれません（私は使ってるけど……）。

＊　＊　＊

手軽でいい音，扱いは簡単，とMIDIをやるには本当にいい時代になったものです。

ただ，GMにしろGSにしろ，プリセットサンプラは扱いやすい代わりに飽きられるのも早いという宿命を持っています。おそらく次世代ではシンセサイザ機能を含んだ共通化音源というのが期待されてくるのでしょう。しかし，最近はPCM音源一辺倒で新しい音源というと物理音源くらいしかないのがちょっとさみしいところですね。

MIDI端子について

以前，WAVE BLASTERの接続記事を読んで，自分でも実際に作ろうとしたときにMIDI端子の信号配置がわからない人がいるというのを聞きました。

MIDIのコネクタは世界的に統一されています。ピン配置は，コネクタを楽器の内部から見たときに，いちばん右が１番ピン，そこから逆時計回りに４，２，５，３と不規則な番号が続きます。このうち，１，３の２本は開放（NC）です（時計の３時，９時の位置のピン）。２番ピンは開放ですが(12時方向のピン)，ケーブルの外側のシールドに接地させなければなりません。

実際の信号線は４番ピンと５番ピンで，それぞれ，１＋，１－の信号となっています。

コネクタを楽器の外から見たときは逆になりますので注意してください。

COMPUTER MUSIC

AD PCMの基礎から応用まで

基本はPCM,あとはみんなついてくる

Nishikawa Zenji 西川 善司

あらゆる音源を模倣できるPCM音源。ここではX68000のPCMによる自在な音の加工とAD PCMによる再生のための基礎知識をまとめてみましょう。その気になれば実に多彩な処理が可能になります。

X68000が登場したときは，まだPCM音源っていうのはなんとなく未知の香りが漂っていて，しかも無限の可能性を感じさせる未来的なイメージのある音源だった。

時は流れ，いまでは家庭用ゲーム機器にも当たり前のように装備され，ふと街に出れば，百貨店のエレベータや銀行のキャッシュディスペンサーなどの案内音声もPCMではないか，このままいくとあらゆるものにPCM音源が搭載されるのでは？と私は予測する。そのうち「皮をおむきください」としゃべりだすバナナとか，「散歩につれて行ってください」と切願する犬とかも開発されるのではないかと考えると，怖くて夜も眠れない。

PCMってなんだ？

PCMってのはいったいなんなんだ。どーして数値が音になるんだ？ このあたりをまず簡単に解説しよう。

PCMとはPulse Code Modulationの略記でアナログ信号をデジタル伝送するための変調方式のひとつ。通信とか映像とか幅広い活用がなされている。なにも「音」の信号化のための手段だけじゃない。

原理的にはすごく単純で，入力された信号の強さ(たとえば電圧，電流)を数値に変換する。これが量子化(Quantization)と呼ばれる動作だ。

図1のような0～9までの10段階の数値表現しかできない場合を考える。もし8.3くらいの信号が入力されたとする。でも，表現できるのは0～9だから8.3の入力信号は8.3に一番近い8という数値に変換されてしまう。この血も涙も人情もない処置をデジタル化という。

「サユリ好きなんだ。つきあってくれ」

サユリは鼻毛男がきらいだったとすると，

1)「あんた鼻毛出てるからイヤ」

2)「鼻毛出てるけど優しそうだからまずは友達から」

ということで1)のほうがデジタルってわけ。

同じような手順で，入力された信号をどんどんと数値に変換していって，なにかしらの記憶メディアにそれらを記憶しておく。これを再生する場合は，記録してある数値を読み出して，その値に応じた信号(たとえば電圧，電流)を出力してやればいい。これを復調(Demodulation)という。

さて，先ほどの例で元の信号は8.3だった。だけれどデジタル化して8とされた。復調時にも，記録メディアからもちろん8という数値を読んで8に相当する信号を出力してしまう。ここで元の信号と復調された信号との差は，

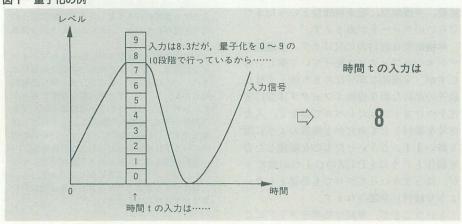
8.3−8.0=0.3

で0.3の原信号との誤差が発生しているよね。これが量子化ノイズ/誤差(Quantization noise/error)と呼ばれるものだ。「音」ならば，この量子化ノイズが大きくなればなるほど原音と違って聞こえてしまうわけだ。

図1では10段階だった記録する解像度の幅をもっと細かくして(広げて)やれば量子化ノイズをなくすことができるんじゃないだろうか。たとえば図1の例で0～9の10段階を0.0，0.1，0.2，0.3…9.9までの100段階にしたらどうだろうか。8.3という入力信号をちゃんと8.3とデジタル化できるではないか。そう，このデジタル化する細かさを量子化レベル/解像度(Quantization level/resolution)といい，ここを，より性能のよいものにすれば原波形をより正確にデジタル化できるようになるのだ(図2)。

よく16ビットPCMとか18ビットオーバーサンプリングとかいって～bitを強調しているけれど，これはその量子化レベルの細かさをいっているんだ。これを特に量子化ビット数(Quantization bit)なんて呼んだりする。「16ビット」ならば0～65535，つまり65536段階の細かさで波形を記録できるというわけ。ビット数が多ければ多いほど性能がよく原信号をより正確に記録/再現できると考えていい。

次に図3を見てほしい。このようにすごい速さで変化を遂げる波形があったとしよう。これを図3(a)のようなのんびりした速度でデジタル化していたのでは，ほら，全然元の波形とは違った形としてデジタル化されてしまうよね。記録間隔をより狭めてやらないと，波形の変化の記録タイミングを見逃してしまうわけ。図3(a)の結果は元の信号とは似ても似つかないカッコになっているから量子化ノイズが大きいどこ

図1 量子化の例

ろかとんでもないことになっちゃってる。入力信号がもし「音」だったとすると再生時の音はすごく汚く聞こえることだろう。

このデジタル化するタイミングを標本化周波数/サンプリング周波数(Sampling frequency)という。これは量子化ビット数とともに原波形を正確に記録し，正確に復元するための2大重要パラメータなのだ。CDは44.1kHz, DATは48kHz, なんてパラメータが有名だけれど，それはそのくらいの細かさでデジタル化したってことを意味する。X68000のAD PCM音源はサンプリング周波数15.6kHz。数値的に見ても現代の標準オーディオの水準から随分遅れを取っているのがわかるなぁ。

こうして見ると，量子化ビット数は横方向の細かさ(解像度)，サンプリング周波数はY方向の細かさを表していることに気づいたかな？

標本化定理(Sampling theorem)ってやつによればサンプリングする波形がW(Hz)を超える周波数成分を含まないとき，2W(Hz)のサンプリング周波数でサンプリングすれば元の波形を完全に復元できる，といっている。人間の耳は大体，下15Hzから上20,000Hz(20kHz)までの音が聞き分けられる。人間が聞ける音はつまり最高周波数成分が20kHzということで，これを正確に記録するためには倍の周波数でサンプリングする必要があるから20kHz×2=40kHzでサンプリングすればいいことになる。ほら，CDのサンプリング周波数の44.1kHzという値に近いでしょ。44.1kHzっていうパラメータには「人間の可聴範囲の音を正確に記録するため」という目的から定められたんだねぇ。

ちょっと工夫したんです∆PCM

先ほど，量子化ビット数っていう話が出てきたっけ。たとえばこのパラメータが16ビットだとするとひとつのデータが占めるデータ量も当然16ビット，つまり2バイトだ。1単位が2バイトだからちょっと長めの信号をこの16ビットPCMで記録したらそのデータ長はとんでもない長さになる。これはなんとかなりませんか，博士？　というわけで考え出されたのが∆PCM(Delta PCM)という方式。

波形っていうのはある短い時間の間に振動している。波形の図を見るとわかるけれどその波形のギザギザの頂点から次の頂点までいく間は多少なりとも時間を食っている。サンプリング周波数が高いときはギザの頂点から頂点へいくまでの変化点を何点もサンプリングすることになる。

そうなると図4のように前のサンプリング時点での値と今回のサンプリング時点の値とではほとんど差がないというケースが多くなる。図4のようにA時点では2134, B時点では2181だとする。16ビットPCMでデータ化すると2134, 2181という2つの値，しめて4バイトが記録される。

でも，前の値とそう大差がないならば現在の値との差だけを記録していけば少ないビット数でデータを記録できるんじゃないの？　と考えだされたのが∆PCMだ。図4の例でいけば2181−2134=47で，47ならば16ビットのデータ長はいらない，8ビット程度で十分だろう，としてしまうわけ。

結局この例では(A時点の前のデータが不明なので計算していないが)算出されるデータは8ビットデータが2つ，つまりPCMのときの半分のデータ量になっている。

本来必要なビット数よりも少ないビット数で記録するのがこの手法の特徴。こうし

図2　量子化解像度

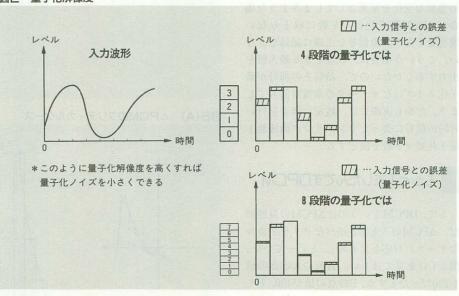
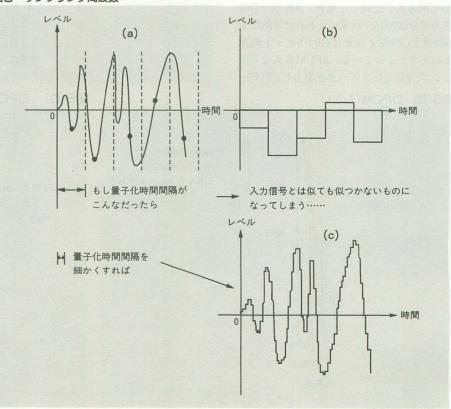



図3　サンプリング周波数

て短いデータ長で上ランクの品質のデータが表現できる！ というわけだ。この前後の値の差を求めることを離散数学では差分(Differential Calculus)という。

復調する場合は0を初期値としてΔPCMデータの値を加算（減算）していき，その都度算出されるPCM値を出力することになる。単純だ。

ただこの方法はサンプリング周波数もある程度高くないと量子化されたデータの質は低くなり効果は上がらない。たとえば図5(A)のようにサンプリング周波数が波形の急激な変化を見過ごしてしまうような場合はΔPCM量子化ビット数に収まらない変化量となり，原信号を正確に記録できない。こういう場合は，表現できる最大値を出力するしかないので，結局その部分が量子化ノイズになりデータの品質は落ちてしまう。だから実際には，厳密に調査を行い目的の信号に合ったサンプリング周波数と量子化ビット数を決定する。

もっと工夫したんですDPCM

さて，DPCMというのはΔPCMの発展形だ。ΔPCMの「先ほど述べたクリティカルなケースに対応したもの」，といっていい。図5(A)を見てほしい。波形のA時点の値が200だったとする，B時点の値が400，そしてC時点が810だとする。これを8ビットΔPCM方式で記録した場合，A時点とB時点の差分は200なので8ビット表記可能だが，B時点とC時点との差は410で8ビット範囲を超えてしまっている。ΔPCM方式ならばここで，255というデータを出力し，原デー

図4 ΔPCMの原理

図5(A) ΔPCMのクリティカルケース

図5(B) DPCM(その1)

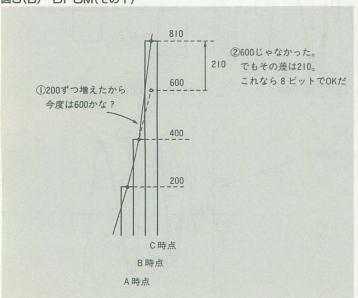
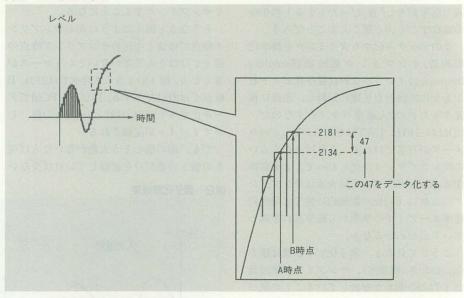
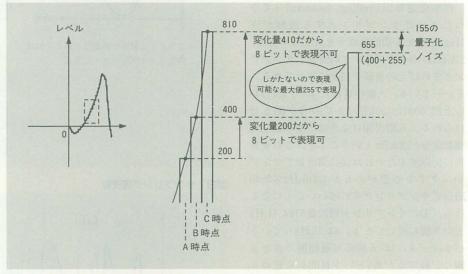
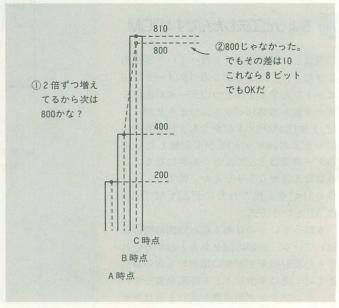



図5(C) DPCM(その2)

タとの410−255＝155の量子化ノイズが発生する。

DPCMではそれまでの変化を学習し(?)，次のPCM値を予測してしまうのだ。図5(B)を参照してほしい。たとえばA時点，B時点と200，400ときているため，あ，こりゃ200ずつ増えているから次はきっと600だなとC時点の値を予測してしまうわけ。これでCの値が810だから，その予測したデータとの差をデータ化する。つまり810-600=210，210ならば，8ビット範囲の値で正しく量子化に成功する。これならばΔPCMの欠点も解決，というわけ。

図5(B)の場合は「差」をパラメータとして次の値を予測したけれど，「比」を予測パラメータとすることもできる。図5(C)がそれ。つまり，A時点，B時点と200，400ときている，フムフム，2倍，2倍ときているから次は400×2＝800に違いないと予測する。でC時点での値は810だから，810−800＝10。ΔPCMでは表現不可能だった変化量が「比」パラメータのDPCMならば10なんていう小さな値に化けてしまうのだ。

究極の工夫かAD PCM

DPCMをさらに工夫したのがAD PCM(Adaptive Differential Pulse Code Modulation)だ。この工夫というのはもはや圧縮の領域に片足踏み入れているといっていいだろう。

DPCMが実際に単位時間あたりにデータ化した値dは，

d＝実際のPCM値−予測したPCM値

だったのは理解できてるよね。AD PCMは次のPCMの値を予測するまではDPCMと同じ。その先がすごい。DPCMでそのまま記録していたdまでも予測するんだ。つまり次のPCM値とDPCMの値の両方を予測していくわけ。

d'＝DPCMの予測値(厳密には絶対値)

とするとこの比率d：d'(つまりd/d')をAD PCMチップ内部のテーブルと参照する。

そのテーブルには，比率の辞書みたいのがあって，実際にそのときの比率にいちばん近い値をその辞書から決定する。それで，その辞書のページをその瞬間のAD PCM値とするわけ。DPCMの値まで予測してしまい，この比率をキーにしてデータ化を行うということは，つまり表現可能なビット数の上限がある意味ではほとんどないのと同じということだ。

極端な話，たとえある瞬間の実測DPCM値が44444444という値だったとしても予測DPCM値が66666666ならば(2/3)という事実がわかり，これでデータ化できてしまうからだ。

データ化を完了したあとは次回の処理に向けて次にくるPCM値とDPCMの予測を行う。以上の動作を図解したものが図6だ。

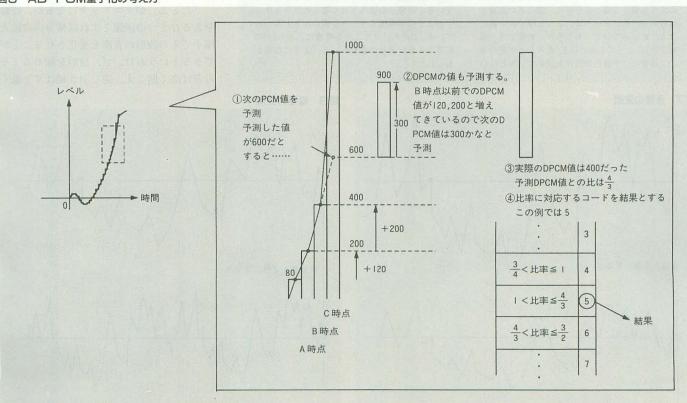
こうしてみるとわかるようにAD PCMの値というのはそのときの辞書の比率の辞書のページなわけだから，AD PCMはいわば音声を圧縮してるイメージだよね。

この，一見するとちょっとしたプログラムにもなってしまうような複雑アルゴリズム&演算処理をX68000のAD PCM音源チップもリアルタイムに行っているのだ。専用ハードっていうのは恐ろしいまでに高速なんだよねぇ。

さて，X68000のAD PCM音源チップの，より具体的な予測アルゴリズムについてはソフトバンク刊「Inside X68000」(桒野雅彦氏著)か1992年Oh!X6月号52ページを参照してほしい。ここではスペースの都合でアルゴリズムのイメージを伝えるだけにとどめることにする。

音のひみつ

ちょっと「音」について考えてみよう。音波というくらいで音は「波」。声帯や楽器といった音源がエネルギーを使って空気を振動させる。この断続的な振動が波として空気を伝わって，やがて人間の鼓膜へたどり着く。鼓膜はその振動を受けて震える。その震えを聴覚細胞が電気的信号に変換して

図6 AD PCM量子化の考え方

脳へ送る。脳はこれを「音」として感じる。音が鳴って聞こえるまでのしくみは大体こんな感じだ。

テープやCDを再生したときはこんな感じだ。結局，音の波形を空気に放出するわけだから，スピーカーのコーン紙が，これに接続された電流に応じて振動する。これで音が生まれる。しかし，コーンがある運動をした場合，そのコーンの動きは慣性の法則などの要因によって，本来の電気信号の指示とは違った動きをしてしまう場合がある。これはコーンの材質や設計に問題がある場合もあるが，このあたりがスピーカーの性能に関係してくるわけ。スピーカーによって音の聞こえ方が違うのはそういった要因が関係している。

音の量子化は要するにその空気振動の波形の振幅を単位時間あたりに記録(数値化)していくことなのだ(厳密的には音波形やその伝わり方は3次元的なものなんだけど)。つまりデジタル化された波形をいろいろいじってしまうとこれを再生したときには音が違って聞こえる。

それでは，この音の波形はどんな特性があるかを見ていこう。

音量を変えるには

スピーカーの音量を上げるときにはツマミを回すよね。このツマミは可変抵抗，つまり電流の量を調節しているツマミ。スピーカーの音量を上げるということは電流の量を調節しているわけ。電流の量でスピーカーのコーンが振動して音波を作り出していることは先ほども述べた通りだから，つまり電流が強ければ強い振動，弱ければ弱い振動ということになる。振動の大きさはつまり振幅の大きさだから……。ここまでいったらもうわかるよね。そう音量は音波形の振幅の大きさそのままなんだ。

ある音の波形まるごと振幅を半分にして鳴らすと物理的音量(＊1)は半分になるというわけ(図7-A)。

ではPCMの次元ではどういうことになるか考えてみよう。

波形を小刻みに数値化していったのがPCMデータだから，振幅を操作するということは，この値そのものを一定比率で大きく小さくしたりすればよいことになる。たとえばある音のPCMデータの構成値の全部を2倍の値にしてしまえばその波形は振幅2倍のものに生まれ変わる。半分にすれば振幅半分の波形に生まれ変わる。結局PCM値の拡大縮小が音量の増減に相当することになる(図7-B)。

(＊1) ここで物理的というのは，人間の耳は物理的音量(音圧という)が大きくなればなるにつれ心理的には小さく聞こえ，音圧が小さければ小さいほど心理的にはより大きく聞こえるという特性を持つ。

音程を変えるには

音程(音高/ピッチ)は波形のギザギザの密度に起因する。単位時間あたりに振動すれば振動するほど，つまりギザギザが多ければ多いほど音は高く聞こえる。逆にゆったりとした振動ならば，つまりギザギザが密集していないような場合は低い音になる。もっとも自然界の音はあらゆる周波数の音，倍音を含んでいるのでいまいったような一意的ないい方が正しいかというとそうではないのだが，イメージ的にはだいたいそんな感じだ。

高い音と低い音の違いはわかった。では，ある音があって，この音の音高を上下させる場合にはどうしたらよいのだろうか。これはその波形をそのまま小さく縮めるか大きく引き延ばすかで実現できる。つまり波形のギザギザの密度を上げるか下げるかの処理を行うわけだ。

「たとえ」がマニアックになるが，波形をあるひとつの画像とすれば横方向の拡大縮小でその波形の音高を変化させることができるというわけ。で，波形を縮めるとその音は高く聞こえ，逆に引き延ばすと低く

音色ってなんだろう

音色の違いは参考文献の「音楽用語の基礎知識」によれば倍音成分をどれだけどんな割合でどんなパターンで含んでいるかによるとある。

同書によれば低い倍音が比較的強いとホルンのような豊かで幅のある音に，逆に高い倍音が比較的強ければオーボエのような硬く鋭い音になる，といっている。もちろん，音量変化(振幅変化)によっても音の硬さ/柔らかさが変わるので倍音だけが音色の特性を決めているとはいい切れないが。

倍音というのは大きく分けて奇数倍音と偶数倍音というのがある。これは基音となる周波数成分の偶数倍であるか，奇数倍であるかをいっている。同書は奇数倍が強く響くとクラリネットのように不均衡でうつろな音になるともいっている。このあたりのことを考慮に入れればFM音源でも思い通りの音が作れる(ようになるかもしれない)。

図7 音量の変換

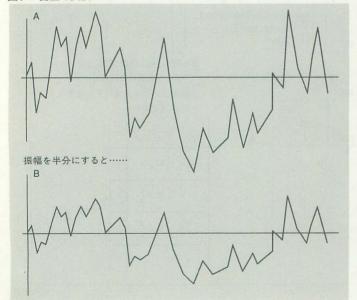
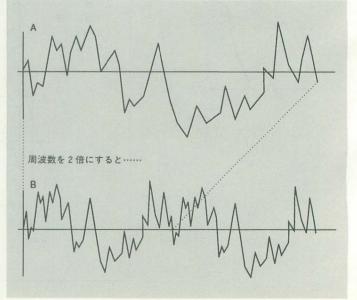



図8 音程の変換

なる(図8-A)。

では次にPCMの次元で考えてみよう。PCMはPCMデータの値をある時間間隔で読み出しその値に応じた出力を行っている。だから，この出力する時間間隔を狭める，つまり出力速度を速くすれば波形は縮まるのは理解できるだろう。ということはPCM出力速度を速くすれば音程は上がることになる。同様に逆に遅くすれば音は低くなるのだ。これを図解すると図8-Bのようになる。

和音を作るには

2つの音が同時に鳴った場合は波形はどうなっているのだろう。

実は答えは簡単。図9のように単純に波形が重なりあっているだけなのだ。

PCMの次元で考えるとどうなるか。2つの波形を重ねあわせるのだから，単純にそれぞれの音のPCMデータを加算していくだけだ。

ここで，いくつかの変わった例について考えてみよう。

もし，まったく同じPCMデータ同士を加算したらどうなるか。それぞれのPCM値が2倍になるので結局振幅が2倍の音ができあがっちゃう(図10A)。

それではもし，絶対値が同じで符号の違う，まったく逆位相の波形同士を加算したらどうなるか。そう，あらゆるPCM値は0になってしまい結局無音になってしまう(図10B)。

現在この技術を利用した「ノイズバスター」という電子耳栓が発売されている。外界の音をサンプリングし，リアルタイムに逆位相の波形を耳へ送ることにより，結局耳には振幅0の音が聞こえる(→音が聞こえない)というものだ。

なんだかここまでわかってくると音って簡単なオモチャに見えてくるよね。

AD PCMは面倒くさい

さて，ここまででPCMデータ(音波形)と実際に聞こえる「音の特徴」の関係が理解できたと思う。この知識はそのままAD PCMやDPCMに適用できるのだろうか。

結論からいえばできるわけがない。

なぜならAD PCMデータは前述したように入力された波形をそのまま数値化せずに賢いアルゴリズムのもとに一風変わった数値化によって生まれたものである。いわばAD PCMデータは音波を圧縮暗号化して記録したものなわけだ。だから，AD PCMデータそのものを前節まで述べたような手法で加工しても希望どおりの効果は得られない。

しかし，暗号は暗号化したアルゴリズムを逆に利用して解読してやれば，元の音の波形に戻すことができるはず。実際，AD PCM音源ではこの圧縮暗号化されたデータを随時展開演算してPCMデータを取得，そのPCMデータ値に応じた出力をしているのだ。

そういうわけでAD PCMデータ音を加工する場合，まず一度AD PCM化したときのアルゴリズムを逆利用して，AD PCMからPCMを逆算し，一度，ピュアなPCMデータにすることから始めなければならない。

図9　和音変換の考え方

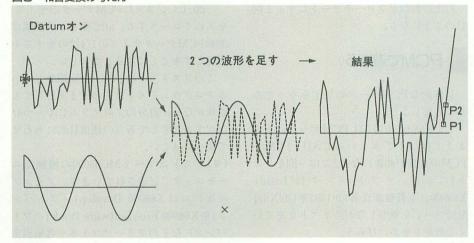



図10.A　同じ波形同士を足す

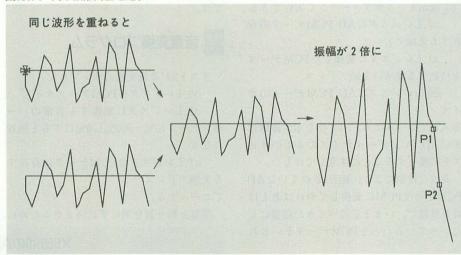
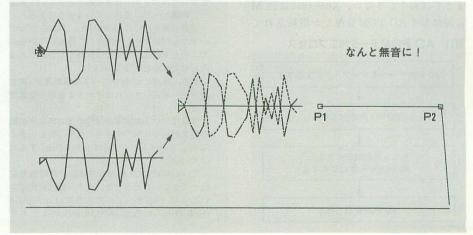



図10.B　振幅の絶対値が同じで符号が逆の波形を重ねると

PCMデータになったところで，加工処理を施す。そして，加工されてできたPCMデータはそのままではAD PCM音源では演奏できないから，音を記録したとき，AD PCM音源チップが行ったAD PCM化アルゴリズムをエミュレートしてAD PCM化する。

これでやっと生まれ変わったAD PCMデータの完成だ。

以上の動作をフローチャートにすると図11のようになる。

PCMで遊ぼう

具体的なPCMデータの加工をやってみることにしよう。

だがX680x0用のAD PCMデータはそのままでは加工できないのでAD PCM→PCMの変換が必要になる。この一例をリスト1に示す。ソフトバンク刊「Inside X68000」(桒野雅彦氏著)か1992年Oh!X6月号52ページを参照しながらリストを見ていくとわかりやすいだろう。

AD PCM→PCM変換はルーチンadpcm_to_pcmというサブルーチンで実行できる。

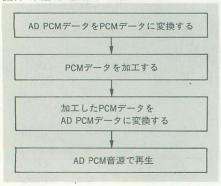
a0.LレジスタにAD PCMデータの存在する先頭アドレス

a1.Lレジスタに変換したPCMデータを格納する領域の先頭アドレス

d0.LレジスタにAD PCMデータのサイズ

を入れてコールする。a1で指し示す領域はd0のAD PCMデータサイズの4倍のサイズを必要とすることに注意してほしい。

高度な技術により圧縮(?)されていたAD PCMを一度PCMに変換してやればあとは好き放題だ。いままで述べてきた蘊蓄にしたがっていろいろとPCMデータをいじれる。

いじったPCMデータはX680x0ではそのままでは再生できない。X680x0にはPCM音源はなくAD PCM音源しか搭載されていないからだ。

そこで，今度は逆にPCM→AD PCM変換という作業が必要になる。これの一例は同じくリスト1に示した。ルーチンpcm_to_adpcmがそれだ。

a0.Lレジスタに変換後のAD PCMデータを格納する領域の先頭アドレス

a1.LレジスタにPCMデータの存在する先頭アドレス

d0.LレジスタにPCMデータのサイズを入れてコールする。a0で指し示す領域はd0のPCMデータサイズの1/4倍のサイズを必要とすることに注意してほしい。

このリスト1をはじめ，今回紹介しているプログラムリストはそのまま実行しても意味がない。自分のプログラムにくっつけたりするなどして各自の使用目的にあわせて使ってほしい。

(※)リストはすべてMC68000の機械語ニーモニックで記述されています。アセンブル実行にはX68000 Develop(ソフトバンク)やX68000 Free Software Book(ソフトバンク)などのフリーソフト本や電脳倶楽部別冊1号に収録されているHAS.Xが必要です。

音量変換プログラム

リスト2が音量変換のサンプルだ。

d0.LレジスタにPCMデータサイズ

d1.Lレジスタに変換する音量のパーセンテージ(0%~999%)。0%にすると無音になる

a1.LレジスタにPCMデータの存在する先頭アドレス

でコールする。

演算を割り算を用いずに済ませるために，一度パーセンテージパラメータを128分解能のパラメータへとリスト先頭で変換している。目的のPCM値を128分解能パーセンテージと積算し，128で除算している(7ビット右へシフトで実現)。

音量の変更されたPCMデータは変換元のPCMデータの存在したアドレスへ上書きされることに注意。

音程変換プログラム

リスト3が音程変換のサンプルだ。

d4.LレジスタにPCMデータのサイズ

d5.Lレジスタに変換後のデータを格納するために確保した領域の最終アドレス

d6.Wレジスタに変換元の音の周波数(1~65535Hz)

d7.Wレジスタに変換後の周波数(1~65535Hz)

a1.Lレジスタに変換後のPCMデータを格納する領域の先頭アドレス

a2.Lレジスタに変換元のPCMデータが格納されている先頭アドレス

でコールする。戻り値としてd0.Lに変換後のPCMデータのサイズを返す。

音程の変換の解説のところで「波形の画像を拡大縮小するイメージ」と述べたがこのプログラムではまさにその処理を行っている。周波数aの音のPCMデータの長さがmで，これを周波数bに変換するということはPCMデータの長さを(a/b)×mにすることにほかならない。

(b/a)×mじゃないの？　と思う人もいるかもしれない。しかし周波数はデータ長の逆数に比例するのでこれでよいのだ。データ長を時間(周期)Tとみなせば周波数fは，

X680x0のAD PCM音源

「AD PCM音源ていうのは音程と音量は変化させることはできない音源なんだよ」という間違った認識が一部に浸透しているようである。

実際の発声時にはAD PCMチップはAD PCMデータからPCMデータへ変換して再生しているのでこのときに工夫を施せばPCMでできることはみんなできるといっていい。

まず，単純にAD PCMチップの駆動速度(再生周波数)を変化させればそのまま音程が変化できる。

音量については実際の信号の出力時にその絶対値を適当な比率で大きくしたり(音が大きくなる)，小さくしたり(音は小さくなる)するだけでいい。

ではX680x0のAD PCM音源はなぜ音程音量変化ができないのだろうか。まず，音程変化についてはAD PCMチップへの入力クロックが3.9kHz,5.2kHz,7.8kHz,10.4kHz,15.6kHzの5とおりしかないため，5とおりの音程変化しかできないのだ。もし，この入力クロックがなんらかの形で任意可変だったなら自由に音程変化をさせることができただろう。音量変化はチップにそういう機能がなかった……というだけのこと。

しかし，ソフトウェアの力でリアルタイムにADPCM→PCM→音量変化→ADPCMのプロセスを実行するAD PCMドライバが開発された。そう，PCM8.X／江藤啓氏作がそれだ。PCM8.XはAD PCM→PCM変換時に複数の疑似チャンネル管理を行い，PCM同士の加算処理で最大8和音のパラレル発声を可能にしている。

さらに最近では，AD PCM→PCM変換したデータを音量疑似複数チャンネル管理処理に加えて，指定されたパラメータに応じて間引いたり補間したりして音程までもパラレルに変化させることのできるPCM8.Xの上位機能バージョンRDN.X/ゆ～きい！作が発表されている。

図11　AD PCMデータ加工プロセス

f=1/T

で求められることは物理の時間に習っただろう。これで周波数とデータ長の関係は明白。ということで周波数を2倍するときはデータ長は1/2，逆に周波数を1/2するとデータ長は2倍にすることになる。

まだわからない人のためにもうちょっと具体的に説明しよう。440Hzの音を220Hzの音にする場合，音は220Hzのほうが音程が低いよね。音程が低いということは波長が長くなる(波形が長くなる)ということだったでしょ。つまりその音の波形を2倍長く延ばすということになるわけだ。この例ではほら周波数が1/2，PCMのデータ長は2倍。とこうなるわけ。以下数学的帰納法(?)で同様だ。

で，このプログラムではPCMデータ長を変換元と変換後の周波数の比の「逆数」になるように変換しているだけ。波形を縮めるときは元の波形のPCM値の一部を省略して切り詰め，引き延ばす場合は線形補間演算で求められた予測PCM値を埋め込んでいく(図12)。

ピッチベンド

音程変換ができるなら，この音程変換動作を一定でなくて線形的に連続的に行っちゃえ，ってのがリスト4だ。

リスト3では，初めに周波数変換の「比の逆数」パラメータをプログラムの先頭で求め，あとはこのパラメータに従ってデータを切り詰めたり，引き延ばしたりしているけれど，このパラメータをだんだんと変化させてやるわけ。元の周波数から目的の周波数までどんどん周波数が変化していくということは，この2つの値の「比の逆数」もどんどん変化するということだからね。演算回数はリスト3に比べて格段に増加するので処理速度はちょっと遅い。

なお，このプログラムのパラメータレジスタはリスト3とまったく同じため省略する。

和音変換プログラム

これも原理がわかったらプログラムは簡単だ。2つの音の波形(PCMデータ)がある場合に，これらをミックスしてひとつにしたい場合は，それぞれのPCMデータのPCM値同士を加算すればよい。もちろんPCMデータというのは負の振幅を持った値もあるがこれも加算してしまってOK。この場合は厳密には減算になるが。

この変換のサンプルプログラムをリスト5に示す。

d1.LにPCMデータ1のサイズ

d2.LにPCMデータ2のサイズ

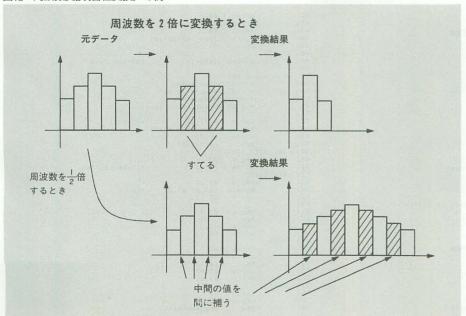
a1.LにPCMデータ1の存在する領域の先頭アドレス

a2.LにPCMデータ2の存在する領域の先頭アドレス

d4.Lにミキシングディレイ

でコールする。そして，

d0.LにミックスされたPCMデータのサイズ

a0.LにミックスされたPCMデータの存在する領域の先頭アドレス

を返す。なお，結果はこのプログラムが自動的に確保したメモリ領域に格納される点に留意してほしい。

ただ2つのPCMデータを足し合わせるだけでは面白くないので，おまけ機能をつけてある。データ1とデータ2をミックスする場合にミキシングディレイを設定できるような仕様がついている(図13)。

d4.Lレジスタパラメータがこの機能のパラメータだ。このディレイを0にすればデータ1とデータ2は同時に鳴るようなデータが作成される。このディレイに値を与えればまずデータ2が鳴り，このディレイの値の時間だけやや遅れてデータ1が鳴る。

もし，2つの音の音量の比率を変えてミックスする場合は，初めにそれぞれの音量を音量変換プログラムで任意の音量に変換し，それからそれぞれをミキシングすればよい。それぞれを違った音程でミキシングしたい場合も同様だ。

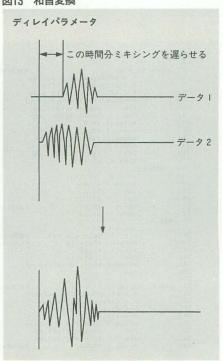
音量変換処理は実は簡単に和音変換プログラムに組み込むことが可能だ。このあたりは自由研究ということにしておこう。

広がるPCMの可能性

一度PCMのデータにしてしまうとその音声データはとたんに柔軟ないろんな可能性を持ったものになる。

Oh!X1991年12月号で石上達也氏が紹介した「FFT(高速フーリエ変換)」をPCMデータに対して行えば周波数成分に分解する

図12　周波数変換(音程変換)の例

図13　和音変換

ことができる。このデータをいじって倍音成分を変えればまったく別の音を作れる(かもしれない)。

また，1994年1月号で私が紹介した「畳み込み演算」を行えば本格的なデジタルエフェクト処理も可能だ。

もちろんなにもそんな数学的な高度な知識や技術を用いなくてもPCMデータの加工は可能だ。最後に紹介するのはギターのエフェクトで有名なディストーション処理。人の声にかけたりすると電話の声みたいになる。これは音量変換処理の応用で簡単に実現できる。もちろん実際のエフェクタは各エフェクタメーカー特有の技術を導入して，個性ある音色を醸し出してくれるが，基本原理は今回紹介するものと大差ないはずだ。

まず，入力波形の振幅を任意の倍率で拡大する。これはつまり音量増加の処理。これに振幅の上限と下限を設定したフィルタに通してやればいい。この処理を図解すると図14のようになる。そしてこれをプログラムにしたのがリスト6だ。

d0.LにPCMデータのサイズ

d1.wに倍率パラメータ(正数で0～32767)

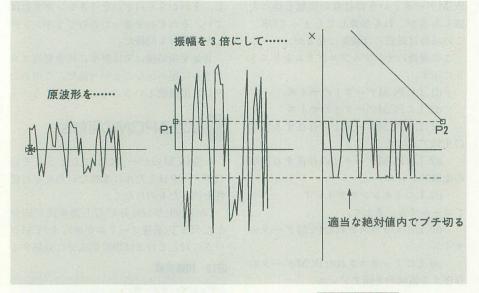
d2.Lに上限下限レベルの絶対値(正数で0～32768)

a1.LにPCMデータの存在する領域の先頭アドレス

を入れてコールする。音量変換プログラムのときと同じく変換結果のPCMデータは変換元のPCMデータの存在したアドレスへ上書きされることに注意。

倍率パラメータはPCMデータ値と直接掛け合わすもので，だいたい実用上，10～60程度が適当だ。上限下限レベルはX68000で再生することを考えるならば100～500前後が適当だろう。まぁいろいろ試してみてほしい。

ご覧のとおり，実に単純明快な処理だ。このほか，コーラスやディレイをはじめいろんなエフェクトが，このように単純な演算で実行できる。アイデア次第で君オリジナルのエフェクトだって作れるかもしれないぞ。また，この手のプログラムは高度なプログラム技術を必要としないからアセンブラや機械語の勉強にももってこいだ。

図14 ディストーション

PCMはおもしろい

さて音とPCMデータの関係のイメージが理解できたかな。もういまや(AD)PCM音源は「音をデジタル録音」するだけの手段で満足してちゃいけない。これからはその先，「録音した音をいかに応用するか」までできているといえる(ホントか)。

X680x0のAD PCM音源は現在では我々ユーザーが他機種のPCM音源のスペックに嫉妬を覚えてしまうような状況になってしまったけれども，まだまだ遊べる。これを機会にAD PCM音源と戯れてみよう。

参考文献

遠藤三郎，「音楽用語の基礎知識」，シンコーミュージック

桒野雅彦，「Inside X68000」，ソフトバンク

リスト1

```
 1: *       list1
 2:
 3: adpcm_to_pcm:                     *ADPCM→PCM変換
 4:         * < a0=adpcm data buffer
 5:         * < a1=pcm data buffer
 6:         * < d0.l=adpcm data size
 7:         movem.l d0-d7/a0-a6,-(sp)
 8:         lea     scaleval(pc),a5
 9:         lea     levelchg(pc),a6
10:         moveq.l #0,d3
11:         moveq.l #0,d7
12:         moveq.l #$0f,d4
13:         add.l   d0,d0
14:         lea     last_val(pc),a3
15:         move.w  d3,(a3)
16: __atp_lp:
17:         move.b  (a0),d1
18:         and.w   d4,d1
19:         tst.b   d4
20:         bpl     @f
21:         lsr.b   #4,d1             *get 4bit data
22:         addq.w  #1,a0
23: @@:
24:         not.b   d4
25:         bsr     calc_pcm_val      *実際の計算
26:         move.w  d2,(a1)+          *add pcm data to buffer
27:         subq.l  #1,d0
28:         bne     __atp_lp
29:         movem.l (sp)+,d0-d7/a0-a6
30:         rts
31:
32: pcm_to_adpcm:                     *PCM→ADPCM変換
33:         * < a0=adpcm data buffer
34:         * < a1=pcm data buffer
35:         * < d0.l=pcm data size
36:         * - all
37:         movem.l d0-d7/a0-a6,-(sp)
38:         lea     scaleval(pc),a5
39:         lea     levelchg(pc),a6
40:         moveq.l #0,d6             *scalelevel=0
41:         moveq.l #0,d7
42:         moveq.l #0,d4
43:         lsr.l   d0
44:         lea     last_val(pc),a3
45:         move.w  d4,(a3)           *clr.w  (a3)
46: pta_lp:
47:         move.w  (a1)+,d3          *d3=pcm data
48:
49: calc_adpcm_val:
50:         * < d3.w=pcm value
51:         * < d7.w=scale level
52:         * > d1.b=adpcm value
53:         * > d7.w=next scale level
54:         * X
55:         sub.w   (a3),d3           *d3=diff
56:         bmi     case_diff_minus
57:         moveq.l #0,d1
58:         bra     calc_diff
59: case_diff_minus:
60:         neg.w   d3
61:         moveq.l #8,d1             *d1:become data
62: calc_diff:
63:         add.b   d7,d7
64:         move.w  (a5,d7.w),d2      *=d
65:         lsr.b   d7
66:
67:         cmp.w   d3,d2
68:         bge     _or2
69:         sub.w   d2,d3
70:         ori.b   #4,d1
71: _or2:
72:         lsr.w   d2
73:         cmp.w   d3,d2
74:         bge     _or1
75:         sub.w   d2,d3
76:         ori.b   #2,d1
77: _or1:
78:         lsr.w   d2
79:         cmp.w   d3,d2
80:         bge     chg_scalelvl
```

▶6月号は初心者の私にもわかりやすかったのでよかった。　上田　利三(20)兵庫県

```
 81:         ori.b   #1,d1
 82:
 83: chg_scalelvl:
 84:         add.b   d1,d1
 85:         add.w   (a6,d1.w),d7     *scalelevl+=levelchg(adpcm value)
 86:         bmi     rst_sclv_
 87:         cmpi.w  #48,d7
 88:         bls     mk_olv
 89:         moveq.l #48,d7
 90:         bra     mk_olv
 91: rst_sclv_:
 92:         moveq.l #0,d7
 93: mk_olv:
 94:         exg     d7,d6
 95:         bsr     calc_pcm_val_
 96:         exg     d6,d7
 97: *-------------------------------------------------------------------
 98:         not.b   d4
 99:         bne     set_lower
100:                                  *case upper 4bits
101:         lsl.b   #3,d1
102:         or.b    d1,d5
103:         move.b  d5,(a0)+
104:         bra     check_cnt
105: set_lower:
106:         lsr.b   d1
107:         move.b  d1,d5
108: check_cnt:
109:         subq.l  #1,d0
110:         bne     pta_lp
111:         movem.l (sp)+,d0-d7/a0-a6
112:         rts
113:
114: calc_pcm_val:
115:         * < d1.b=adpcm value
116:         * < d7.w=scale level
117:         * > d2.w=pcm value
118:         * > d7.w=next scale level
119:         * > d1.b=adpcm*2
120:         * < a3.l=last_val
121:         * X d3 d2
122:         add.b   d1,d1
123: calc_pcm_val_:
124:         add.b   d7,d7
125:         move.w  (a5,d7.w),d3     *=d
126:         lsr.b   d7
127:
128:         move.w  cpv(pc,d1.w),d2
129:         jmp     cpv(pc,d2.w)
130: abc:
131:         add.w   (a3),d2
132: *       bsr     chk_ovf
133:         move.w  d2,(a3)          *d2=pcmdata
134:
135:         add.w   (a6,d1.w),d7     *scalelevl+=levelchg(adpcm value)
136:         bmi     rst_sclv
137:         cmpi.w  #48,d7
138:         bls     allend
139:         moveq.l #48,d7
140: allend:
141:         rts
142: rst_sclv:
143:         moveq.l #0,d7
144:         rts
145:
146: cpv:
147:         dc.w    cpv0000-cpv
148:         dc.w    cpv0001-cpv
149:         dc.w    cpv0010-cpv
150:         dc.w    cpv0011-cpv
151:         dc.w    cpv0100-cpv
152:         dc.w    cpv0101-cpv
153:         dc.w    cpv0110-cpv
154:         dc.w    cpv0111-cpv
155:         dc.w    cpv1000-cpv
156:         dc.w    cpv1001-cpv
157:         dc.w    cpv1010-cpv
158:         dc.w    cpv1011-cpv
159:         dc.w    cpv1100-cpv
160:         dc.w    cpv1101-cpv
161:         dc.w    cpv1110-cpv
162:         dc.w    cpv1111-cpv
163:
164: cpv0000:
165:         lsr.w   #3,d3
166:         move.w  d3,d2
167:         bra     abc
168: cpv0001:
169:         lsr.w   #2,d3
170:         move.w  d3,d2
171:         lsr.w   d3
172:         add.w   d3,d2
173:         bra     abc
174: cpv0010:
175:         lsr.w   d3
176:         move.w  d3,d2
177:         lsr.w   #2,d3
178:         add.w   d3,d2
179:         bra     abc
180: cpv0011:
181:         lsr.w   d3
182:         move.w  d3,d2
183:         lsr.w   d3
184:         add.w   d3,d2
185:         lsr.w   d3
186:         add.w   d3,d2
187:         bra     abc
188: cpv0100:
189:         move.w  d3,d2
190:         lsr.w   #3,d3
191:         add.w   d3,d2
192:         bra     abc
193: cpv0101:
194:         move.w  d3,d2
195:         lsr.w   #2,d3
196:         add.w   d3,d2
197:         lsr.w   d3
198:         add.w   d3,d2
199:         bra     abc
200: cpv0110:
201:         move.w  d3,d2
202:         lsr.w   d3
203:         add.w   d3,d2
204:         lsr.w   #2,d3
205:         add.w   d3,d2
206:         bra     abc
207: cpv0111:
208:         move.w  d3,d2
209:         lsr.w   d3
210:         add.w   d3,d2
211:         lsr.w   d3
212:         add.w   d3,d2
213:         lsr.w   d3
214:         add.w   d3,d2
215:         bra     abc
216: cpv1000:
217:         lsr.w   #3,d3
218:         move.w  d3,d2
219:         neg.w   d2
220:         bra     abc
221: cpv1001:
222:         lsr.w   #2,d3
223:         move.w  d3,d2
224:         lsr.w   d3
225:         add.w   d3,d2
226:         neg.w   d2
227:         bra     abc
228: cpv1010:
229:         lsr.w   d3
230:         move.w  d3,d2
231:         lsr.w   #2,d3
232:         add.w   d3,d2
233:         neg.w   d2
234:         bra     abc
235: cpv1011:
236:         lsr.w   d3
237:         move.w  d3,d2
238:         lsr.w   d3
239:         add.w   d3,d2
240:         lsr.w   d3
241:         add.w   d3,d2
242:         neg.w   d2
243:         bra     abc
244: cpv1100:
245:         move.w  d3,d2
246:         lsr.w   #3,d3
247:         add.w   d3,d2
248:         neg.w   d2
249:         bra     abc
250: cpv1101:
251:         move.w  d3,d2
252:         lsr.w   #2,d3
253:         add.w   d3,d2
254:         lsr.w   d3
255:         add.w   d3,d2
256:         neg.w   d2
257:         bra     abc
258: cpv1110:
259:         move.w  d3,d2
260:         lsr.w   d3
261:         add.w   d3,d2
262:         lsr.w   #2,d3
263:         add.w   d3,d2
264:         neg.w   d2
265:         bra     abc
266: cpv1111:
267:         move.w  d3,d2
268:         lsr.w   d3
269:         add.w   d3,d2
270:         lsr.w   d3
271:         add.w   d3,d2
272:         lsr.w   d3
273:         add.w   d3,d2
274:         neg.w   d2
275:         bra     abc
276:
277: scaleval:
278:         dc.w     16,17,19,21,23,25,28
279:         dc.w     31,34,37,41,45,50,55
280:         dc.w     60,66,73,80,88,97,107
281:         dc.w     118,130,143,157,173,190,209
282:         dc.w     230,253,279,307,337,371,408
283:         dc.w     449,494,544,598,658,724,796
284:         dc.w     876,963,1060,1166,1282,1411,1552
285: levelchg:
286:         dc.w     -1,-1,-1,-1,2,4,6,8
287:         dc.w     -1,-1,-1,-1,2,4,6,8
288:
289: last_val:        ds.w    1
```

リスト2

```
 1: *        list2
 2:
 3: level_chg:                         *パーセンテージを16進変換
 4:         * < d0=pcm data size
 5:         * < d1=percentage(0%~999%)
 6:         * < a1=pcm data address
 7:         movem.l d0-d2/a1,-(sp)
 8:         lsr.l   d0                 *d0=data count
 9:         lsl.l   #7,d1              *d1=d1*128
10:         divu    #100,d1            *d1=0~999→0~1278
11:         andi.l  #$ffff,d1          *(128を基準としたパーセンテージ)
12: lvch_lp:
13:         move.w  (a1),d2
14:         muls    d1,d2              *音量演算
15:         asr.l   #7,d2              *音量演算
16:         move.w  d2,(a1)+           *store pcm data to buffer
17:         subq.l  #1,d0
18:         bne     lvch_lp
19:         movem.l (sp)+,d0-d2/a1
20:         rts
```

▶80MバイトのHDDを使っているが，とにかく少ない。記事を読めば読むほど1GバイトHDDとMOがほしくなった。もっとお金を貯めねば。　古味　貴徳(20)東京都

リスト3

```
 1: *        list3
 2:
 3: pitch_chg:
 4:          * < d4.l=pcm data size
 5:          * < d5.l=mem.block end address
 6:          * < d6.w=source frq
 7:          * < d7.w=destination frq
 8:          * < a1.l=destination pcm data address
 9:          * < a2.l=source pcm data address
10:          * > d0.l=destination pcm data size
11:          movem.l d1-d7/a0-a6,-(sp)
12:          move.l  a1,-(sp)
13:          lsr.l   d4
14:          exg.l   d6,d7
15:          divu    d6,d7
16:          move.w  d7,d1              *d1=step
17:          clr.w   d7
18:          divu    d6,d7
19:          swap    d7
20:          tst.w   d7
21:          beq     @f
22:          add.l   #$0001_0000,d7
23:          clr.w   d7
24: @@:
25:          swap    d7                 *d7=revise
26:          moveq.l #0,d3
27:          tst.w   d1
28:          bne     nz_d1
29: doa_lp00:
30:          move.w  (a2)+,d0
31:          add.w   d7,d3
32:          bcc     @f
33:          move.w  d0,(a1)+
34:          cmp.l   a1,d5
35:          bls     out_of_mem
36: @@:
37:          subq.l  #1,d4
38:          bne     doa_lp00
39:          bra     exit_doa
40: nz_d1:
41:          move.w  d1,d6
42:          move.w  (a2)+,d0
43:          move.w  (a2),d2
44:          cmpi.l  #1,d4              *最後かどうか
45:          bne     @f
46:          move.w  d0,d2
47: @@:
48:          add.w   d7,d3
49:          bcc     @f
50:          addq.w  #1,d6
51: @@:
52:          ext.l   d0
53:          ext.l   d2
54:          movem.l d1/d3,-(sp)
55:          sub.l   d0,d2
56:          divs    d6,d2
57:          move.w  d2,d1              *d1=step
58:          clr.w   d2
59:          divu    d6,d2
60:          swap    d2
61:          tst.w   d2
62:          beq     @f
63:          add.l   #$0001_0000,d2
64:          clr.w   d2
65: @@:
66:          swap    d2                 *d2=revise
67:          moveq.l #0,d3
68: doa_lp01:
69:          move.w  d0,(a1)+
70:          cmp.l   a1,d5
71:          bls     out_of_mem
72: @@:
73:          add.w   d1,d0
74:          add.w   d2,d3
75:          bcc     @f
76:          tst.w   d1
77:          bpl     doa_pls
78:          subq.w  #1,d0
79:          bra     @f
80: doa_pls:
81:          addq.w  #1,d0
82: @@:
83:          subq.w  #1,d6
84:          bne     doa_lp01
85:          movem.l (sp)+,d1/d3
86:          subq.l  #1,d4
87:          bne     nz_d1
88: exit_doa:
89:          sub.l   (sp)+,a1
90:          move.l  a1,d0
91:          movem.l (sp)+,d1-d7/a0-a6
92:          rts
```

リスト4

```
  1: *        list4
  2:
  3: pitch_bend_chg:
  4:          * < d4.l=pcm data size
  5:          * < d5.l=mem.block end address .
  6:          * < d6.w=source frq
  7:          * < d7.w=destination frq
  8:          * < a1.l=destination pcm data address
  9:          * < a2.l=source pcm data address
 10:          * > d0.l=destination pcm data size
 11:          movem.l d1-d7/a0-a6,-(sp)
 12:          lea     work(pc),a6
 13:          move.l  a1,-(sp)
 14:          lsr.l   d4
 15:          exg.l   d6,d7
 16:          move.l  d6,atb_frqsrc-work(a6)
 17:          move.l  d6,atb_frqnow-work(a6)
 18:                                     *周波数変化率計算
 19:          move.l  d4,d1
 20:          move.l  d7,d0
 21:          sub.l   d6,d0
 22:          bpl     @f
 23:          neg.l   d0
 24:          bsr     wari               *d0.l/d1.l=d0.l...d1.l
 25:          neg.l   d0
 26:          move.l  d0,atb_step-work(a6)        *周波数変化率
 27:          move.l  #-1,atb_sgn-work(a6)
 28:          bra     .atb0
 29: @@:
 30:          bsr     wari               *d0.l/d1.l=d0.l...d1.l
 31:          move.l  d0,atb_step-work(a6)        *周波数変化率
 32:          move.l  #1,atb_sgn-work(a6)
 33: atb0:
 34:          swap    d1
 35:          clr.w   d1
 36:          move.l  d1,d0
 37:          move.l  d4,d1
 38:          bsr     wari               *d0.l/d1.l=d0.l...d1.l
 39:          tst.l   d1
 40:          beq     @f
 41:          addq.l  #1,d0              *revise
 42: @@:
 43:          move.w  d0,atb_rvs-work(a6)
 44:
 45:          bsr     calc_frqchgrate
 46:
 47:          moveq.l #0,d3
 48:          move.w  d3,atb_rvswk-work(a6)
 49:
 50:          tst.b   atb_sgn-work(a6)
 51:          bpl     nz_d1_b
 52: doa_lp00_b:
 53:          move.w  (a2)+,d0
 54:          add.w   d7,d3
 55:          bcc     @f
 56:          move.w  d0,(a1)+
 57:          cmp.l   a1,d5
 58:          bls     out_of_mem
 59: @@:
 60:          bsr     calc_frqchgrate
 61:          subq.l  #1,d4
 62:          bne     doa_lp00_b
 63:          bra     exit_doa_b
 64: nz_d1_b:
 65:          move.w  d1,d6
 66:          move.w  (a2)+,d0
 67:          move.w  (a2),d2
 68:          cmpi.l  #1,d4              *最後かどうか
 69:          bne     @f
 70:          move.w  d0,d2
 71: @@:
 72:          add.w   d7,d3
 73:          bcc     @f
 74:          addq.w  #1,d6
 75: @@:
 76:          ext.l   d0
 77:          ext.l   d2
 78:          movem.l d1/d3,-(sp)
 79:          sub.l   d0,d2
 80:          divs    d6,d2
 81:          move.w  d2,d1              *d1=step
 82:          clr.w   d2
 83:          divu    d6,d2
 84:          swap    d2
 85:          tst.w   d2
 86:          beq     @f
 87:          add.l   #$0001_0000,d2
 88:          clr.w   d2
 89: @@:
 90:          swap    d2                 *d2=revise
 91:          moveq.l #0,d3
 92: doa_lp01_b:
 93:          move.w  d0,(a1)+
 94:          cmp.l   a1,d5
 95:          bls     out_of_mem
 96: @@:
 97:          add.w   d1,d0
 98:          add.w   d2,d3
 99:          bcc     @f
100:          tst.w   d1
101:          bpl     doa_pls_b
102:          subq.w  #1,d0
103:          bra     @f
104: doa_pls_b:
105:          addq.w  #1,d0
106: @@:
107:          subq.w  #1,d6
108:          bne     doa_lp01_b
109:          movem.l (sp)+,d1/d3
110:          bsr     calc_frqchgrate
111:          subq.l  #1,d4
112:          bne     nz_d1_b
```

▶秋葉原の某店が出しているパソコン雑誌のレーダーチャートでOh!Xはマニア度，内容レベルが非常に高い値だった。ちなみに内容レベルは確かトップだったと思う。これからもハイクオリティを保ってほしい。
栗原　将史(26)神奈川県

```
113: exit_doa_b:
114:          sub.l   (sp)+,a1
115:          move.l  a1,d0
116:          movem.l (sp)+,d1-d7/a0-a6
117:          rts
118:
119: calc_frqchgrate:                             *変換パラメータの計算
120:          movem.l atb_frqsrc(pc),d6-d7
121:          add.l   atb_step(pc),d7
122:          move.w  atb_rvs(pc),d1
123:          add.w   d1,atb_rvswk-work(a6)
124:          bcc     @f
125:          add.l   atb_sgn(pc),d7
126: @@:
127:          move.l  d7,atb_frqnow-work(a6)
128:
129:          divu    d6,d7
130:          move.w  d7,d1               *d1=step
131:          clr.w   d7
132:          divu    d6,d7
133:          swap    d7
134:          tst.w   d7
135:          beq     @f
136:          add.l   #$0001_0000,d7
137:          clr.w   d7
138: @@:
139:          swap    d7                  *d7=revise
140:          rts
141:
142: wari:                                        *32ビット/32ビット=32ビット...32ビット
143:          * < d0.l/d1.l=d0.l ...d1.l
144:          cmpi.l  #$ffff,d1
145:          bls     divx                *16ビット以下の数値なら1命令で処理
146:          cmp.l   d0,d1
147:          beq     div01               *d0=d1商は1
148:          bls     div02               *1命令では無理なケース
149:
150:          move.l  d0,d1               *商は0余りはd0.l
151:          moveq.l #0,d0
152:          rts
153: div01:                                       *商は1余り0
154:          moveq.l #1,d0
155:          moveq.l #0,d1
156:          rts
157: div02:
158:          movem.l d3-d5,-(sp)
159:          move.l  d1,d3
160:          clr.w   d3
161:          swap    d3
162:          addq.l  #1,d3
163:          move.l  d0,d4
164:          move.l  d1,d5
165:          move.l  d3,d1
166:          bsr     divx
167:          move.l  d5,d1
168:          divu    d3,d1
169:          divu    d1,d0
170:          andi.l  #$ffff,d0
171: div03:
172:          move.l  d5,d1
173:          move.l  d5,d3
174:          swap    d3
175:          mulu    d0,d1
176:          mulu    d0,d3
177:          swap    d3
178:          add.l   d3,d1
179:          sub.l   d4,d1
180:          bhi     div04
181:          neg.l   d1
182:          cmp.l   d1,d5
183:          bhi     div05
184:          addq.l  #1,d0
185:          bra     div03
186: div04:
187:          subq.l  #1,d0
188:          bra     div03
189: div05:
190:          movem.l (sp)+,d3-d5
191:          rts
192: divx:
193:          movem.w d0/d3,-(sp)
194:          clr.w   d0
195:          swap    d0
196:          divu    d1,d0
197:          move.w  d0,d3
198:          move.w  (sp)+,d0
199:          divu    d1,d0
200:          swap    d0
201:          moveq.l #0,d1
202:          move.w  d0,d1
203:          move.w  d3,d0
204:          swap    d0
205:          move.w  (sp)+,d3
206:          rts
207:
208: work:
209: atb_step:        ds.l     1        *autobend work
210: atb_rvs:         ds.w     1        *autobend work
211: atb_rvswk:       ds.w     1        *autobend work
212: atb_frqsrc:      ds.l     1        *autobend work
213: atb_frqnow:      ds.l     1        *autobend work
214: atb_sgn:         ds.l     1        *autobend work
```

リスト5

```
 1: *        list5
 2:
 3: mix_pcm:               *2つのPCMデータをミキシングする
 4:          * < d1.l=size of pcm data1
 5:          * < d2.l=size of pcm data2
 6:          * < a1.l=address of pcm data1
 7:          * < a2.l=address of pcm data2
 8:          * < d4.l=mixing delay(offset) of pcm data1
 9:          * > a0.l=destination pcm data address
10:          * > d0.l=destination pcm data size
11:          movem.l d1-d7/a1-a6,-(sp)
12:          move.l  d1,d3
13:          add.l   d4,d3
14:          cmp.l   d2,d3
15:          bhi     @f
16:          move.l  d2,d3               *d2の方が大きかったら
17: @@:
18:          move.l  d3,d4
19:          move.l  d3,-(sp)
20:          DOS     _MALLOC
21:          addq.w  #4,sp
22:          tst.l   d0
23:          bmi     out_of_mem
24:          move.l  d0,a0               *a0=destination address
25:
26:          movem.l d2/a1-a2,-(sp)
27:          move.l  d3,d2               *size
28:          movea.l a2,a1               *source
29:          movea.l a0,a2               *dest
30: @@:
31:          move.w  (a1)+,(a2)+
32:          subq.w  #2,d2
33:          bne     @b
34:          movem.l (sp)+,d2/a1-a2
35:
36:          move.l  d4,d3
37:          sub.l   d2,d3               *お尻の部分をクリア
38:          beq     go_mix
39:          lea     (a0,d2.l),a3
40:          moveq.l #0,d0
41: @@:
42:          move.w  d0,(a3)+            *いっぺんに2つクリア
43:          subq.l  #2,d3
44:          bne     @b
45: go_mix:
46:          *a1=use parameter
47:          move.l  d4,d0                            *size
48:          lea     (a0,d4.l),a2
49: @@:
50:          move.w  (a1)+,d2
51:          add.w   (a2),d2
52:          move.w  d2,(a2)+
53:          subq.l  #2,d1
54:          bne     @b
55:          movem.l (sp)+,d1-d7/a1-a6
56:          rts
```

リスト6

```
 1: *        list6
 2:
 3: distortion:                    *ディストーション処理
 4:          * < d0.l=pcm data size
 5:          * < d1.w=magnification parameter
 6:          * < d2.l=cutoff level
 7:          * < a1.l=pcm data address
 8:          movem.l d1/d3-d4/a1,-(sp)
 9:          lsr.l   d0                  *d0=data count
10:          tst.w   d1
11:          bpl     @f
12:          neg.w   d1                  *d1は正数
13: @@:
14:          tst.l   d2
15:          bmi     @f
16:          neg.l   d2                  *d2を負数にする
17: @@:
18:          move.l  d2,d3
19:          neg.l   d3                  *d3は正数の上限(d2は負数の下限)
20: dis_lp:
21:          move.w  (a1),d4
22:          muls    d1,d4               *倍率演算
23:          cmp.l   d2,d4               *下限チェック
24:          bge     @f
25:          move.w  d2,(a1)+
26:          bra     next_dis
27: @@:
28:          cmp.l   d3,d4               *上限チェック
29:          ble     @f
30:          move.w  d3,(a1)+
31:          bra     next_dis
32: @@:
33:          move.w  d4,(a1)+
34: next_dis:
35:          subq.l  #1,d0
36:          bne     dis_lp
37:          movem.l (sp)+,d1/d3-d4/a1
38:          rts
```

▶6月号の表紙って，すごくいいですね。捻られた文字に鉄格子。一瞬見ただけでは「なんてダークなのだろう」と思うだけですけれどね。深い。　　堂領　輝昌(20)神奈川県

16ビットPCMデータを加工する

黙っていたけどスゴイぞZPLK

Tateno Mitsuru 舘野 暢

COMPUTER MUSIC

Z-MUSICシステムに含まれる大量の16ビットPCMデータ。ZPLKを使えばそれらを活用してさまざまなデータの加工が行えます。最終出力はAD PCMでも，肝心の計算は16ビット精度ですので高音質が保てます。

お久し振りです。前の特集以来ちゃんとZ-MUSICerしてましたか？

前回はZPCNVの使用例を挙げましたね。今回はタイトルの通りZPLKを取り上げます。サンプルリストをいくつか実行したあとでZ-MUSIC用データを再生してみたいと思います。

ZPLKってなに？

さてこのツール。Z-MUSICシステムver.2から正式にリリースされたモノです。いったいどんなモノでしょう。前回のテーマだったZPCNVはコンバータでしたね。LKってくらいだからリンカでしょ？　という人が多数でしょうね……。そう，当たりです。簡単にいえば数あるPCMファイルをつなげます（最大32個直結）。

たとえばAとBのPCMがあったなら，ABBBBBBBBのようにスマートにつなげられます。ZPCNVだとどうしても2行以上のコマンドリストになってしまうでしょうね。そのあたりからしても結構キレイにコマンドが書けます。

PCM同士をつなげる前の理屈

X68000のPCMはAD PCMですね。データの形式としては前のデータとの差分形式です。AD PCMを2つも3つもつなげると音量が高くなったりしませんか？

それはAデータの最後にBをつなげたとすると，前のデータからの差分を取ってしまう都合で音量が大きくなったり小さくなったりします。差分の宿命のようなものです。

このままでは当初の目的である「キレイにAD PCMをつなぐ」が実現できません。ではどうしましょう。困りましたね。うーん。ここはひとつZ戦士隊長に相談しましょうか。

善「16ビットPCMなら可能なんだな。ブチッ，NO CARRIER」

おおぅ，回線が切れた。うちのモデムは部屋の蛍光灯を消すとよく落ちるのでした……。悲しい。

悲しいけど16ビットなら平気そうです。それは0を基準とする符号付きデータ形式だからだそうです。

ポイントはいつでも0が基準というところでしょうね。この先のデータも基準が決まっているからいくつ並べられても平気という利点がある感じですね。しかもデータのクオリティが高いのです。

要するにAD PCMの差分形式の場合は前のデータがある意味で基準だからマズイ，つなげるためには16ビットPCMに変換してからということになります。

おわかりですか？　おかわりは不可。

つなげてループしてみましょう

ところでZ-MUSICのディスク5の中身を見てください。ディレクトリの中にはファイルの拡張子が「.P16」というものがたくさんありますね。ディスクラベルにも書いてあるようにこのディスクには16ビットPCMがツメ込まれています。

それでは例として，DG_P16の中のDG D2.P16とDG_D2LP.P16をつないでみましょう。話の都合上，DG_D2.P16のほうがA，DG_D LP.P16をBとします。ではABBBBBBBBというようなつなぎ方をしてみましょうかね。

マニュアルの45ページを見るとスイッチによっていろいろと加工できることがわかりますね。ここでは「つなぐ」ですから，えーと，そう，リンク制御の「-X」。

書式は，

zplk A -Xl,r,t B ［出力ファイル名］

ですね。細かくなるけどループタイプは「0のそのまま」。Bの「ループ回数は8回」。リンク作業前には0の「なにも行わない」と設定しておきます。

実際には，

zplk DG_D2.P16 -X0,5,0 DG_D2LP.P16 ABBBBBBBB.P16

というふうにすれば新たな16ビットPCMができるはずです。試してみてください。

うまく加工が終了したなら196562バイトのファイルができたはずです。それにしてもデカいですね。

16ビットPCMはAD PCMよりも4倍ほど容量を取ります。これは16ビットPCMのデータの都合上しかたないことです。あとから，.P16→.PCM変換をすればとりあえずサイズの問題は解決しますから我慢してください。変換後は当然1/4になります。

加工された16ビットPCMを聞くには「-A」スイッチによる.P16→.PCM変換が通常は必要です。しかし，実は黙ってましたがファイルのAD PCM変換加工をしなくても再生できます。ウムをいわずPCM8を組み込んで，

zplk -A ABBBBBBBB.P16 PCM

としてみてください。ちゃんとAとB×8がつながっているでしょ？

つなげることのみじゃないのよ(重要)

実はオイラ，初めの頃はZPLKって「つなげることしかできない」と若干バカにしていたところがあったのですが，いま考えると頭をボーズにしたい気持ちでいっぱいです（一部転載禁止）。

ZPLKの能力をさらりと紹介すると，まず，AD PCMから16ビットPCMに変換。その逆の変換。範囲内カッティング。音量を増加減。フェードイン・アウト。リンク・ループ。周波数変換。ポルタメント。畳み込み演算。逆転再生。こういったことがスイッチによりできます（一覧参照）。

「畳み込み演算」は普通聞きなれない言葉

でしょうね。1994年1月号で西川氏が説明してますが，実は内部的なことはオイラ自身もよくわかってないのです。マニュアルによると「インパルスデータを与えることにより，任意のデータに対してホールの残響効果やボコーダー処理など」とありますね。ふむふむ。

ただし畳み込み演算はデータが大きくなればなるほどメモリと計算時間がものすごく多くかかります。詳しくは1月号をどうぞ。

時間の都合上初めは小さめなデータから始めるとよいでしょうね。それでも面白いものに仕上がっていると思いますよ。

サンプルテクニック

今回はそのＺＰＬＫのデモということでまたしてもサンプルデータを作りました。「Oh! X LIVE in '94」だけでなく，これもよろしくね。ZMSデータのほうは1992年6月号の進藤氏のデータをパワーアップした感じでしょうか。ほとんど音色から作り直しましたがMMLは参考にさせてもらいました。この場を借りてお礼を申し上げます。

それから肝心のバッチファイルがたくさんあります。入力はリスト1から順にしてくださいね。

おおまかなリスト＆テクの解説ですが，

□リスト1

これはリンクの作業です。さっきも説明したからあえて飛ばします。

□リスト2

えーと，おおぅ，いきなり上等テクです。ここでは周波数変換という名の下で実はFM音源でいう「ディチューンずらし」をやっています。微妙に音程をズラすことによりPCMでも同様の効果が実現できます。9行と10行の変換先周波数の値が10違うでしょう。違う周波数のPCMをミックスさせることによってできるというわけです。効果大なのでぜひ活用してください。

なお，このバッチファイルではファイルをミックスするのにZVTを使ってます。ご注意を。

□リスト3

リスト2で作られた.PCMをまた.P16に変換し，ポルタメントを多用して奇妙なことをしています。どんなPCMになるかはバッチが終了してのお楽しみ。ただし数値は正しく入力してくださいよ。違う数値だと間違いなくマヌケになります。

ちなみにこの数値はマニュアルの46ページの「12平均率音階周波数表」やテンポと音長の関係を計算して出る数値です。数値の調整は結構たいへん。

□リスト4

ここもリスト2と同じように周波数変換したあとにディチューン効果を出しています。元の入力ファイルはdgm_d3.pcmで，disk4の¥GUITARに入ってます。

□リスト5

任意の音程からポルタメントの上げ下げをやっています。

＊　＊　＊

うまくできましたか？　できないとオイラのミスかな(おい)。すべてうまくバッチが通れば8つの.PCMファイルができているハズです。暇があれば再生してみてもよいでしょう（PCM8常駐時など）。

copy ［filename］ PCM

です。

補足ですが，強者な人なら気づいているかもしれませんが，2つの16ビットPCMのミキシングはいまのところZVTのコマンドモードでは対応していません（ビジュアルモードならできる)。次バージョンから対応予定だそうです。

やっと聞く準備が整う

とりあえずこのデータを使って曲を演奏する人は，さらにSHAKE2.CNF，SHAKE2_DG.CNFの2つを入力してZPCNVで2つの.ZPDを作成してください。スーパーハングオンのPCMを用意することも忘れずにね。

zpcnv SHAKE2_DG.CNF

zpcnv SHAKE2.CNF

実際に聞くには SHAKE2.ZMSを頑張って打ち込んで，

zp SHAKE2.ZMS

ですね。

サンプルデータでは納得がいかない人は(こら)，またじっくり読み返し＆作り直して実験するのもよい方法でしょう。たいていこういうモノは自分で試すのがいちばんですからね。

ただし前回に続き調子にのって自分の声をサンプリングして，頑張って畳み込み演算して留守電なんかに使っちゃうと，電話がかかってきてもメッセージが入ってない可能性大なので十分気をつけるようにね。

では，またいつか音楽特集で会えたらいいな。

リスト1

```
 1:
 2: echo off
 3: echo  List1.BAT
 4: echo リンクのサンプル。
 5: echo この結果 dg_d.p16と dg_a.p16が出来る。
 6: echo 「Shake the Street (c)SEGA」の為に / TTN
 7: echo on
 8:
 9: zplk dg_d2.p16 -x0,5,0 dg_d2lp.p16 dg_d.p16
10: zplk dg_a2.p16 -x0,5,0 dg_a2lp.p16 dg_a.p16
```

リスト2

```
 1:
 2: echo off
 3: echo  List2.BAT
 4: echo  周波数変換と2つのファイルの混ぜ合せのサンプル。
 5: echo この結果 _dg_a2.pcmと _dg_d2.pcmが出来る。
 6: echo 「Shake the Street (c)SEGA」の為に / TTN
 7: echo on
 8:
 9: zplk dg_a.p16 -t1000,0994 -v10 -a _1.pcm
10: zplk dg_a.p16 -t1000,0984 -v45 -a _2.pcm
11: zvt _1.pcm _2.pcm _dg_a2.pcm -m
12: del _2.pcm > NUL
13: del _1.pcm > NUL
14: zplk dg_d.p16 -t1000,0994 -v10 -a _1.pcm
15: zplk dg_d.p16 -t1000,0984 -v45 -a _2.pcm
16: zvt _1.pcm _2.pcm _dg_d2.pcm -m
17: del _2.pcm > NUL
18: del _1.pcm > NUL
```

リスト3

```
 1:
 2: echo off
 3: echo  List3.BAT
 4: echo  .PCM→.P16の変換と ポルタメントのサンプル。
 5: echo この結果 _dg_d3wave.pcmが出来る。
 6: echo 「Shake the Street (c)SEGA」の為に / TTN
 7: echo on
 8:
 9: zplk _dg_a2.pcm -p _dg_a2.p16
10: zplk _dg_d2.pcm -p _dg_d2.p16
11:
12: zplk _dg_d2.p16 -b29366,27718,2523,7569 -t29366,27718,10093 _1.p16
13: zplk     _1.p16 -b27718,29366,10093,3785 -t27718,29366,13879 _2.p16
14: del _1.p16 > NUL
15: zplk     _2.p16 -b29366,27718,13879,7569 -t29366,27718,21449 _3.p16
16: del _2.p16 > NUL
17: zplk     _3.p16 -b27718,29366,21449,3785 -t27718,29366,25235 _4.p16
18: del _3.p16 > NUL
19: zplk     _4.p16 -b29366,28000,25235,7569 -t29366,27718,32805 _5.p16
20: del _4.p16 > NUL
21: zplk     _5.p16 -b27718,29366,32805,3785 -t28000,29366,36591 -a _dg_d3wave.
pcm
22: del _5.p16 > NUL
```

リスト4

```
 1:
 2: echo off
 3: echo  List4.BAT
 4: echo  周波数変換のサンプル。
 5: echo この結果 _dgm_d3.pcmが出来る。
 6: echo 「Shake the Street (c)SEGA」の為に / TTN
 7: echo on
 8:
 9: zplk dgm_d3.pcm -p dgm_d3.p16
10: zplk dgm_d3.p16 -t0997,1010 -v09 -a _1.pcm
11: zplk dgm_d3.p16 -t0997,1000 -v42 -a _2.pcm
12: zvt _1.pcm _2.pcm _dgm_d3.pcm -m
13:
14: del dgm_d3.p16 > NUL
15: del _2.pcm > NUL
16: del _1.pcm > NUL
17: del dg_a.p16 > NUL
18: del dg_d.p16 > NUL
```

リスト5

```
 1:
 2: echo off
 3: echo  List5.BAT
 4: echo  周波数変換の変換とポルタメントのサンプル。
 5: echo この結果 _dg_b3down.pcmと _dg_b2down.pcmと
 6: echo            _dg_b_2down.pcmと _dg_a2down.pcmが出来る。
 7: echo 「Shake the Street (c)SEGA」の為に / TTN
 8: echo on
 9:
10: zplk _dg_d2.p16 -t29366,49388 _dg_b3.p16
11: zplk _dg_a2.p16 -t22000,24696 _dg_b2.p16
12:
13: zplk _dg_b3.p16 -b49388,24696,34062,11354 -t49388,24696,45417 -a _dg_b3down
.pcm
14:
15: zplk _dg_b2.p16 -b24696,18500,41416,41416 -t24696,18500,82833 -a _dg_b2down
.pcm
16: zplk _dg_b2.p16 -t24696,23308,19870 -a _dg_b_2down.pcm
17: zplk _dg_a2.p16 -t22000,20700,17031 -a _dg_a2down.pcm
18: del _dg_d2.p16 > NUL
19: del _dg_a2.p16 > NUL
20: del _dg_b3.p16 > NUL
21: del _dg_b2.p16 > NUL
22:
23: echo List1から List5までのバッチを実行すると8つの .PCMが出来る。
```

リスト6

```
 1:
 2: / SHAKE2.CNF by MIYA0348 TTN
 3: / ZPCNV SHAKE2.CNF
 4:
 5: / 注意:1993/8月号 P113 の「SUPER HANGON ADPCM分解」ENG氏作を
 6: /         実行してから「ZPCNV SHAKE2.CNF」して下さい。
 7:
 8:
 9: 0       = FCK.PCM,v12
10: .o2c    = FCK.PCM,v60,m0,512
11: 0       = SHPS.PCM,v45,p2
12: .o3d    = RvBS1.PCM,v75,p6,m0
13:
14: 1       = SPH_CH.PCM,v07
15: 2       = SPH_CH.PCM,v30,m1,512
16: .o4f    = SPH_CH.PCM,v130,m2,512
17: .o4f+   = .o4f,v070
18:
19: .o5a    = SPH_CC.PCM,v180
20: .o5a+   = CRASH11.PCM,v70,p-1
21:
22:
23: 3       = SPH_LT.PCM,p-2,v10
24: 2       = SPH_LT.PCM,p-2,v20,m3,522
25: 1       = SPH_LT.PCM,p-2,v35,m2,522
26: .o3g    = SPH_LT.PCM,v160,p-2,m1,522
27:
28: 3       = SPH_MT.PCM,v23,p-2
29: 2       = SPH_MT.PCM,v33,p-2,m3,522
30: 1       = SPH_MT.PCM,v63,p-2,m2,522
31: .o3a    = SPH_MT.PCM,v200,p-1,m1,522
32:
33: 3       = SPH_HT.PCM,v10,p-2
34: 2       = SPH_HT.PCM,v20,p-2,m3,522
35: 1       = SPH_HT.PCM,v35,p-2,m2,522
36: .o3b    = SPH_HT.PCM,v120,p-1,m1,522
37:
38:
39: 4       = CARDR.PCM,v110,p-9
40: .o5g    = CARDR.PCM,v110,p-9,m4,540
41: .o5g+   = .o5g,v70
42: 3       = K808.PCM,v40
43: .o5b    = SLAP1.PCM,v75,p1,m3
44:
45:
46: .erase 4
47: .erase 3
48: .erase 2
49: .erase 1
50: .erase 0
```

リスト7

```
 1:
 2: / SHAKE2_DG.CNF by MIYA0348 TTN
 3: / ZPCNV SHAKE2_DG.CNF
 4:
 5:
 6: 0       = _DG_A2.PCM
 7: 1       = _DG_D2.PCM
 8: 2       = _DGM_D3.PCM
 9:
10:
11: .adpcm_bank 2
12: 3       = 2,p-10,c100,3000,v50
13: .o3e    = 2,p-10,c100,3000,m3,440
14: 3       = 2,p-6,c80,3000,v50
15: .o3a-   = 2,p-6,c80,3000,m3,400
16: 3       = 2,p-5,c50,3000,v50
17: .o3a    = 2,p-5,c50,3000,m3,400
18: 3       = 2,p-4,c30,1500,v50
19: .o3b-   = 2,p-4,c30,1500,m3,400
20: 3       = 2,p-3,c15,3000,v50
21: .o3b    = 2,p-3,c15,3000,m3,400
22: 3       = 2,p-1,c6,1500,v50
23: .o4d-   = 2,p-1,c6,1500,m3,400
24: 3       = 2,    c0,3000,v50
25: .o4d    = 2,    c0,3000,m3,400
26: 3       = 2,p1, c0,1500,v50
27: .o4e-   = 2,p1, c0,1500,m3,400
28: 3       = 2,p2, c0,1500,v50
29: .o4e    = 2,p2, c0,1500,m3,400
30: 3       = 2,p4, c0,1500,v50
31: .o4g-   = 2,p4, c0,1500,m3,400
32: 3       = 2,p6, c0,1500,v50
33: .o4a-   = 2,p6, c0,1500,m3,400
34: 3       = 2,p7, c0,1500,v50
35: .o4a    = 2,p7, c0,1500,m3,400
36: .erase 3
37:
38:
39: .adpcm_bank 3
40: .o3a-   = 0,p-1,c16,1500
41: .o3a    = 0,c14,23000
42: .o3b-   = 0,p1,c0,1500
43: .o3b    = 0,p2,c0,24200
44: .o4d-   = 1,p-1,c0,23000
45: .o4d    = 1,c0,23000
46: .o4e-   = 1,p1,c0,1500
47: .o4e    = 1,p2,c0,24200
48: .o4g-   = 1,p4,c0,1500
49: .o4a    = 1,p7,c0,11500
50:
51:
52: .adpcm_bank 4
53: .o3a    = _dg_a2down.pcm,c14,12000
54: .o3b-   = _dg_b_2down.pcm,c0,12000
55: .o3b    = _dg_b2down.pcm,c0,25000
56: .o4d    = _dg_d3wave.pcm,c0,15000
57: .o4b    = _dg_b3down.pcm,c0,15000
58:
59:
60: .erase 2
61: .erase 1
62: .erase 0
```

リスト8


```
 1:
 2: .comment -TURBO OUTRUN- Shake the Street (C)SEGA  by TTN 94
/05/24(暫定)(+PCM8)
 3:
 4: / SHAKE2.ZMS for ZMUSICsystem 内蔵音源版
 5: / 要PCM8 ZMUSIC ZPCNV ZPLK ZP Zbooks
 6:
 7: / Programmed by 舘野 暢 / MIYA0348 TTN 94/05/24     7/6は誕生日
 8: /                                                     さかな元気?
 9: / ライブニュース
10: /    ZMUSICをサポートする正式ネット 開局準備中   埼玉県大宮市
11: /    援助求む(爆)
12:
13:
14: (i)
15: (o176)
16:
17: / OPM
18: (m1,2000)(aFm1,1)
19: (m2,2000)(aFm2,2)
20: (m3,2000)(aFm3,3)
21: (m4,2000)(aFm4,4)
22: (m5,2000)(aFm5,5)
23: (m6,2000)(aFm6,6)
24: (m7,2000)(aFm7,7)
25: (m8,2000)(aFm8,8)
26: / ADPCM
27: (m9 ,2000)(aAdpcm, 9)
28: (m10,2000)(aAdpcm,10)
29: (m11,2000)(aAdpcm,11)
30: (m12,2000)(aAdpcm,12)
31: (m13,2000)(aAdpcm,13)
32: (m14,2000)(aAdpcm,14)
33: (m15,2000)(aAdpcm,15)
34: (m16,2000)(aAdpcm,16)
35:
36: .ADPCM_BLOCK_DATA=SHAKE2.ZPD
37: .ADPCM_BLOCK_DATA=SHAKE2_DG.ZPD
38:
39:
40: (@2,  31,  4,  1,  0,  2, 36,  0,  6,  3,  0,  0
41:       31, 10,  5,  5,  2,  5,  0,  2,  3,  0,  0
42:       31,  4,  1,  0,  1, 39,  0, 14,  7,  0,  0
43:       21, 10,  5,  5,  2,  5,  0,  2,  7,  0,  0
44:        4,  7, 15,  3)
45:
46: (@4,  31,  6,  2,  7, 12, 24,  0,  1,  0,  0,  0
47:       31,  3,  0,  7,  2,  7,  1,  2,  0,  0,  0
48:       31,  3,  0,  7,  2,  7,  1,  0,  0,  0,  0
49:       31,  3,  0,  7,  2,  7,  1,  1,  0,  0,  0
50:        5,  7, 15,  3)
51:
52: (@5,  23,  0,  0,  5,  0, 28,  0,  0,  0,  0,  0
53:       23,  0,  0,  7,  0,  0,  0,  1,  0,  0,  0
54:       23,  0,  0,  7,  0,  4,  0,  0,  0,  0,  0
55:       23,  0,  0,  7,  0,  4,  0,  0,  0,  0,  0
56:        5,  7, 15)
57:
58: (@9 , 31,  8,  6,  7,  2, 33,  2,  6,  2,  0,  0
59:       25,  5,  6,  7,  1, 53,  1,  5,  2,  0,  0
60:       31,  7,  6,  6,  1, 33,  1,  0,  2,  0,  0
61:       31,  6,  6,  7,  2,  0,  1,  1,  2,  0,  0
62:        0,  6, 15,  3)
63:
64:
65: /----------------
66: /          初期状態
67: (t1)  @9 o2 L8 q8    v16 @k-2 @q48
68: (t2)  @4 o5 L8 q8    v12 @k-1
69: (t3)  @4 o5 L8 q8    v12 @k-2
70: (t4)  @5 o5 L8 q8    v12 @k-1
71: (t5)  @5 o5 L8 q8 p2 v09 @k-2
72: (t6)  @5 o5 L8 q8    v12 @k-1
73: (t7)  @2 o5 L4 q8    v12 @k00
74: (t8)  @2 o5 L4 q8 p2 v12 @k00 y15,0
75:
76: (t1)  r1    [do]
77: (t2)  r1    [do]
78: (t3)  r1    [do]
79: (t4)  r1    [do]
80: (t5)  r1r8  [do]
81: (t6)  r1    [do]
82: (t7)  r1    [do]
83: (t8)  r1r*4 [do]
84:
85:
86: /----------------
87: /         OPM Part
```

▶6月号の特集を見てMOがほしくなっちゃったよー! タブレットもいいし，これは夏のボーナスだなあ。
藍原　和久(23)東京都

```
 88: (t1)  @9 o2 L8 q8    v16 @k-2 @q48
 89: (t1)  eer4.e-rer^2q7e4.<r1r2@q48d>b<e->b
 90: (t1)  |:3|:8e<e>:||:4a<a>:||:4b<b>:|:|
 91: (t1)  |:16e<e>:|
 92: (t1)  |:  |:|:12e<e>:|e<ee-eg-ed->b |:8e<e>:||:8b<b>:|
 93: (t1)  |:4a<a>:||:4b<b>:||:8e<e>:|:|
 94: (t1)  |:|:8d<d>:||:8e<e>:|:|<|:8d-<d->:|>|:8a<a>:|
 95: (t1)  |:12b<b>:|||:4b<b>:| :|r1
 96: (t1)  _6eerrrreerrrrrer8~3d4dddddddddddde-_3
 97: (t1)  _6eerrrreerrrred-r8~3d4dddddddd-dd-dd-dd-d~3
 98: (t1)  |:4|:8e<e>:||:8d<d>:|:|
 99: (t1)  |:|:8d<d>:||:8e<e>:|:|<|:8d-<d->:|>|:8a<a>:||:12b<b>:
|r1
100:
101:
102: (t2)  @4 o5 L8 q8    v12 @k-1 r*768 r*3072
103: (t3)  @4 o5 L8 q8    v12 @k-2 r*768 r*3072
104: /
105: (t2)  |:L8|:@4o5r |:
106: (t2)  g+&_20p1r16&~3r16r~17p3 g+&_20p1r16&~3r16r~17p3
107: (t2)  a4ag+&_20p1r16&~3r16~17p3
108: (t2)  f+| g+&_20p1r16&~3r16r4~17p3 :|e&_20p1r16&~3r16r4~17p
3
109: (t2)  g+&_20p1r16&~3r16r~17p3 g+&_20p1r16&~3r16r~17p3
110: (t2)  a4ag+&_20p1r16&~3r16~17p3
111: (t2)  f+g+&_20p1r16&~3r16~17p3 f+2.^8 e4d+1
112: (t2)  ee&_20p1r16&~3r16~17p3 d+&_20p1r16&~3r16~17p3
113: (t2)  e&_20p1r16&~3r16~17p3 f+4f+&_20p1r16&~3r16~17p3
114: (t2)  e&_20p1r16&~3r16~17p3 f+&_20p1r16&~3r16~17p3 g+*408:|
115: /
116: (t3)  |:L8|:@4o5r |:3 e&_20p2r16&~3r16r~17p3
117: (t3)  e&_20p2r16&~3r16r~17p3 e4ee&_20p2r16&~3r16~17p3
118: (t3)  ee&_20p2r16&~3r16|r4~17p3 :|~17p3 >b2.^8b4b1
119: (t3)  o5c+|:3c+&_20p2r16&~3r16~17p3 :|d+4d+&_20p2r16&~3r16~
17p3
120: (t3)  c+&_20p2r16&~3r16~17p3  d+&_20p2r16&~3r16~17p3  e*408
 :|
121: /
122: (t2,3)  r*3072 :|
123: /
124: (t2)  @4o5p3L8
125: (t2)  g+g+r2g+g+r2rg+rf+*408
126: (t2)  g+g+r2g+g+r2g+g+rf+*408
127: /
128: (t3)  @4o5p3L8
129: (t3)  eer2eer2rerd*408
130: (t3)  eer2eer2eerd*408
131: /
132: (t2,3)  r*6144
133:
134:
135: (t4)  @5 o5 L8 q8    v12 @k-1 r*672
136: (t4)  |:{>ab<c+d+ef+g+a}2b1&b2b.a.g+e1|f+2:|e4.f+4.
137: (t4)  {e&f+&g+&a&}4 b1&b2b.a.g+e1e2d+2e*768
138: (t5)  @5 o5 L8 q8 p2 v09 @k-2 r*672
139: (t5)  |:{>ab<c+d+ef+g+a}2b1&b2b.a.g+e1|f+2:|e4.f+4.
140: (t5)  {e&f+&g+&a&}4 b1&b2b.a.g+e1e2d+2e*768
141: /
142: (t4)  |:|:r1r1r1r1r1r1
143: (t4)  @5 v13 |:o3b<b<b>>b<<b>>b<<f+b>:| v12
144: (t4)  r1r1 @5o5|:5 v13 eb|<e>:|<d+e v12 :|
145: (t5)  |:|:r1r1r1r1r1r1
146: (t5)  @5p2 |:o3b<b<b>>b<<b>>b<<f+b>:|
147: (t5)  r1r1 @5p2o5|:5 eb|<e>:|<d+e :|
148: (t4)  @5|:o5a1&a1 v11r16
149: (t4)  |:>b&<b&<e&>>b&<<e&>>b&<b&<|b>>:|<<b16 v12 :|
150: (t4)  o5g+1&g+1e1&e1f+1&f+2.e4d+1&d+1  :|
151: (t5)  @5p2|:o5a1&a1  v09r16
152: (t5)  |:>b&<b&<e&>>b&<<e&>>b&<b&<|b>>:| v09 :|
153: (t5)  p3v12o5 L16|:16g+f+ec+:|>~3|:7b<ef+b>:|b<eba>
154: (t5)  |:7b<d+f+b>:|<b<d+f+b&_12r&~3r p2v09
155: (t5)  :|v12
156: /
157: (t4,5) r*1536
158: /
159: (t4)  @5p2o5L1v12 |:b<e2>b2<d&d2{c+dc+c+}2>
160: (t4)  |b<e2>b2<d&d>:|b<e2>b2<f+&f+L8
161: (t4)  @5|:o5a1&a1 v11r16|:>b&<b&<e&>>b&<<e&>>b&<b&<|b>>:|
162: (t4)  <<b16 v12 :|
163: (t4)  o5g+1&g+1e1&e1f+1&f+2.e4d+1&d+1
164: (t5)  @5p1o5L1v09 |:b<e2>b2<d&d2{c+dc+c+}2>
165: (t5)  |b<e2>b2<d&d>:|b<e2>b2<f+&f+L8
166: (t5)  @5p2|:o5a1&a1  v09r16|:>b&<b&<e&>>b&<<e&>>b&<b&<|b>>:
|
167: (t5)  v09 :|
168: (t5)  L16p3v12o5 |:16g+f+ec+:|>~3|:7b<ef+b>:|b<eba>
169: (t5)  |:7b<d+f+b>:|<b<d+f+b&_12r&~3r p2v09
170:
171:
172: (t6)  @5 o5 L8 q8    v12 @k-1 r*768
173: (t6)  |:3e1&e2e.e.e|>a1b1<:|>a1b2b2b*768
174: (t6)  |:|:r*4608
175: (t6)  @5|:o6d1&d1> v13 |:>b&<b&<e&>>b&<<e&>>b&<b&<b>>:|v12:
|
176: (t6)  o6c+1&c+1>a1&a1b*768  :|
177: (t6)  r*1536
178: (t6)  @5p1o5L1|:4g+b2g+2|a&a:|<d&d
179: (t6)  @5L8|:o6d1&d1> v13 |:>b&<b&<e&>>b&<<e&>>b&<b&<b>>:|v1
2:|
180: (t6)  o6c+1&c+1>a1&a1b*768
181:
182:
183: (t7)  @2 o5 L4 q8    v12 @k00 r*3840 |: r*4608
184: (t7)  |:d.e.f+.g+.a<c+>|b.g+.e.e.f+e:|<e.>b.<g+.f+.ed+
185: (t7)  c+2.cc+.d+.e>a2.g+a.b.<c+8>b*792:|
186: (t7)  _3o5L8eer2eer2rer~
187: (t7)  d&|:8{dd}<{dd}>:|
188: (t7)  _o5p3L8eer2eer2g+f+e~
189: (t7)  d&|:8{dd}<{dd}>:|L16
190: (t7)  |:4|:7ee<ee>:|p3eb<e>e
191: (t7)  |:6dd<dd>:|p3da<d>dda
192: (t7)  |<c+d>:|<d>dL8
193: (t7)  o5L4
194: (t7)  |:d.e.f+.g+.a<c+>|b.g+.e.e.f+e:|<e.>b.<g+.f+.ed+
195: (t7)  c+2.cc+.d+.e>a2.g+a.b.<c+8>b*792
196:
197:
198: (t8)  @2 o5 L4 q8 p2 v12 @k00 y15,0 r*3840 |: r*4608
199: (t8)  |:d.e.f+.g+.a<c+>|b.g+.e.e.f+e:|<e.>b.<g+.f+.ed+
200: (t8)  c+2.cc+.d+.e>a2.g+a.b.<c+8>b*792:|
201: (t8)  _3o5L8eer2eer2rer~
202: (t8)  p2d&|:8p2{dd}<p1{dd}>:|
203: (t8)  _o5p3L8eer2eer2g+f+e~
204: (t8)  p2d&|:8p2{dd}<p1{dd}>:|L16
205: (t8)  |:4|:7p2ee<p1ee>:|p3eb<e>e
206: (t8)  |:6p2dd<p1dd>:|p3da<d>dda
207: (t8)  |<c+d>:|<d>dL8
208: (t8)  o5L4p2
209: (t8)  |:d.e.f+.g+.a<c+>|b.g+.e.e.f+e:|<e.>b.<g+.f+.ed+
210: (t8)  c+2.cc+.d+.e>a2.g+a.b.<c+8>b*792
211:
212:
213: /----------------
214: /         PCM Part
215:
216: / H.H.
217: (t9)  @1L4o4v13|:3f:|rv9[do]
218: (t9)  |:16f:|L8|:4|:16ff+:|:|
219: (t9)  |:|:6|:16f|f+:|{f+f+}:|
220: (t9)  |:64ff+:|:|
221: (t9)  |:|:8f+4:||:8f16f+8.:|:|
222: (t9)  |:128ff+:|
223:
224:
225: / BD.
226: (t10) @1L4o4r16v2fff16v9o2c16c4r16[do]
227: (t10) |:12c:||:8c8:|L8|:3|:c4.rcc4rcc4c4c4|r:|c:||:16c4:|
228: (t10) |:|:6c4r4.c4cc4.rcc4r|cc4c4c4cc4.rcc4c:|
229: (t10) c4r4.c4crc4c16c.r4.
230: (t10) |:4c4r4.c4rcc4c4c4r:|
231: (t10) |:c4.rcc4rcc4c4c4r|c4.rcc4rcc4c4c4c:|
232: (t10) |cc4c4c4crc4rc4.r:|cc4r|:4c4:|r4rc
233: (t10) cc4c16c*60cc4r4.rc4c2rcc4rcc4c4c4r
234: (t10) cc4c16c*60cc4r{rcc}4cc4c2rcc4rcc4c4c4r
235: (t10) |:4|:6cc4r:|cc4|c4c4r:|rc4.r
236: (t10) |:4c4r4.c4rcc4c4c4r:|
237: (t10) |:c4.rcc4rcc4c4c4r|c4.rcc4rcc4c4c4c:|
238: (t10) cc4rc4c4c4r2.
239:
240:
241: / SD.
242: (t11) @1L4o3@r1r2.v10d[do]
243: (t11) r2r8v9dd2r4d{dd}8r2..{dd}8{rv6dv7dv8dv9dddd}1
244: (t11) |:24rd:|
245: (t11) v8L8r4.d4.rd16d8.rrd4.rd16d8. rrd4.rrrv6dv7dv8dv9dddd
L4
246: (t11) |:@r1
247: (t11) |:5|:8r|d:|{dd}:|
248: (t11) |:6rd:|dd{dddd}2
249: (t11) |:15rd:|r{dd}
250: (t11) ||:15rd:|{rrdddddd}2
251: (t11) :||:13rd:|rd8{dd}8L16r8dddddddd4r8.@r0
252: (t11) L8r2d2rdd4d2|:4r4|d4:|dd
253: (t11) r2d2@r1v10{dv9dr}4r4@r0d2|:4r4|@r0d4:|d{dd}
254: (t11) |:3|:8r4|d4:|dd:||:7r4d4:|r{ddddd&d}4.L4
255: (t11) |:15rd:|r2
256: (t11) |:13rd:|rd8{dd}8L16r8dddddddd4r8.
257:
258:
259: / CYM.
260: (t12) @1L4o5v9r2.a[do]
261: (t12) a2r8aa8^2r8a.a1a1
262: (t12) a1r*1344a1r*576a1a1a1r2.a+4
263: (t12) |:a1r1|:10a+1r1:|r1r1
264: (t12) a1r1r2.a+4&a+1a1r*576
265: (t12) a1r*576a1r1r1|r2a2:|a1
266: (t12) L8aa2raa2.a1r4.r1L8aa2raa2rb4.a^1r1
267: (t12) |:8a1r1r1|r1:|a1
268:
269:
270: / TOM. etc.
271: (t13) @1L8o3v9r1[do]
272: (t13) r1r1r4.b4aarrrrrrrrr
273: (t13) r*2304
274: (t13) o3rrbrraa o5g+4
275: (t13) o3rbrraa o5g+4
276: (t13) o3rbrraa r4r1
277: (t13) |:r*4584
278: (t13) |:r*360o5g^1|r1:|o3rbbaagg4
279: (t13) r*1152|r1r4.g2rr8:|
280: (t13) r2L16|:r8ag:|r2bbaaggrr
281: (t13) L8r1r4@r1{bagr}4r4.r^1r1@r0
282: (t13) r1@r1{bbaagg}2r*480
283: (t13) r*3072
284: (t13) |:r*360o5g^1|r1:|o3rbbaagg4
285: (t13) r*1152
286: (t13) r2L16|:r8ag:|r2bbaaggrr
287:
288:
289: (t14) @2v918r*191[do]
290: (t14) |:4o4|:16e:|>|:8a:|bb<d->b<e->b<g->b<:|>
291: (t14) |:3e4a-4a4b-b:|eeeeeb<d-e-
292: (t14) |:
293: (t14) |:|:28e:|g-ed->b<|:15e:|>|:15b:|b-b |:7a:||:6b:|ab<
294: (t14) |:17e:|:|
295: (t14) |:>a4a-4a4ba4aa-4a4b4|:e4a-4a4b-b:|:|<
296: (t14) |:d-d-ed-a-a-ed-:|>|:aa<d->a<eed->a:|
297: (t14) |:10b:|<e->b<g-g-e->|:11b:|<e->bbb<d-e-
298: (t14) :|
299: (t14) eer2eer^2erd4|:15d:|eer2eer2a-g-ed4|:15d:|
300: (t14) |:4|:4eeg-e:||:3dded:|aa-g-d:|
301: (t14) |:>a4a-4a4ba4aa-4a4b4|:e4a-4a4b-b:|:|<
302: (t14) |:d-d-ed-a-a-ed-:|>|:aa<d->a<eed->a:|
303: (t14) |:10b:|<e->b<g-g-e->|:11b:|<e->bbb<d-e-
304:
305:
306: (t15) @3v918r*191[do]
307: (t15) o4eer4.e-rer^2e4.a1@4b1@3
308: (t15) |:3eere-re4.rere-re4.>aara-ra4.bbr[illegible].<:|
309: (t15) e1e1e1^1
310: (t15) |:
311: (t15) |:|:3q4eeq8eee-e4e4.e4e-|e4.:|e4 >b8^1^1 @4a^2@3a4
312: (t15) @4b-2.@3b4< e^1^1:|
313: (t15) |:d1^1e4re4.r4 eerere4.:| d-1^1>a1^1b1^1@4b1^1@3<
314: (t15) :|
315: (t15) eer2eer^2erd^1@4d1@3 eer2eer2ed-rd^1rdrd- rdrd
316: (t15) |:e4ere-ere4.ere-ere d4drd-dr |d4.drd-dg-d:|drdrd d-d
e-e&
317: (t15) |:e4ere-ere4.ere-ere d4drd-dr |d4.drd-dg-d:|d4.d4 d-d
g-d
318: (t15) |:d1^1e4re4.r4 eerere4.:| d-1^1>a1^1b1^1b1b8r2..<
319:
320:
321: (t16) @3v218rr48r*191[do]
322: (t16) o4eer4.e-rer^2e4.a1@4b1@3
323: (t16) |:3eere-re4.rere-re4.>aara-ra4.bbrb-rb4.<:|
324: (t16) e1e1e1^1
325: (t16) |:
326: (t16) |:|:3  ee  eee-e4e4.e4e-|e4.:|e4 >b8^1^1 @4a^2@3a4
327: (t16)  @4b-2.@3b4< e^1^1:|
328: (t16) |:d1^1e4re4.r4 eerere4.:| d-1^1>a1^1b1^1@4b1^1@3<
329: (t16) :|
330: (t16) eer2eer^2erd^1@4d1@3 eer2eer2ed-rd^1rdrd- rdrd
331: (t16) |:e4ere-ere4.ere-ere d4drd-dr |d4.drd-dg-d:|drdrd d-d
e-e&
332: (t16) |:e4ere-ere4.ere-ere d4drd-dr |d4.drd-dg-d:|d4.d4 d-d
g-d
333: (t16) |:d1^1e4re4.r4 eerere4.:| d-1^1>a1^1b1^1b1b8r2..<
334:
335:
336: (t1,2,3,4,5,6,7,8) [loop]
337: (t9,10,11,12,13,14,15,16) [loop]
338:
339: (p)
```

▶電子ちゃんの急展開にびっくり。次回が気になる。　大槻　尚義(21)長野県

もっと多彩な音楽表現を

波形メモリを使う

Nakano Shuichi 中野 修一

COMPUTER MUSIC

ちょっとマニュアルを見ただけではよくわからないZ-MUSICの波形メモリ機能。これを使うための最低限の用法から，極限まで活用するためのサポートツールを紹介していきます。

波形メモリとは

Z-MUSIC ver.2.0の目玉機能として波形メモリがサポートされたといっても，実際にそれを活用したデータが現れていないので，いったいどのようなものなのかよくわからない人もいるでしょう。また，名前からだいたいの機能は察しがついてもマニュアルの解説だけでは具体的な使い方のわからないという人も大勢いることと思います。

ここではZ-MUSICに追加された波形メモリ機能の具体的な使い方を示してみたいと思います。

基本原理

波形メモリというのは音源のパラメータを連続的に変えていくためのものです。モジュレーションコマンドが三角波などの簡単な波形に従って音程や音量を自動的に変化させていたのに対し，ユーザーが指定した任意の波形を与えて制御することができるようになります。対象となるのは音量や音程，そしてMIDIでは各種コントロールチェンジに対しても波形メモリを適用することができます。

波形はあらかじめ用意しておきます。それで指定された値がステップタイムごとに音量などのパラメータに加算されていきます。つまり波形メモリを使うというのは与えられた要素の変化やゆらぎを設定することに相当します。

波形の登録はZMSコマンドでは共通コマンドとして用意されており，音楽データの先頭あたりで設定します。たとえば，

```
WAVE_FORM 8,1,16 {
  10,12,15,18,22,15,10,12,
  10, 8, 9, 8, 6, 4, 2, 0,
  -2,-4,-2, 0, 1, 2
}
```

のように設定します。

パラメータは，最初の8が波形番号でユーザーが使用できるのは8～31のものです。

次の1はループタイプを指定するもので す。0，1，2の3種類があり，

0　ループしない
1　最後までいくとループ始点に返る
2　最後までいくと後ろから折り返す

という動作になります。

次の16はループポイントの指定で，16番目の要素から最後までがループの対象になるという意味です。この例でいくと，頭からデータを1回出力し終わると16番目の0のところに戻りそこから最後までの区間のデータを繰り返して出力するようになります。

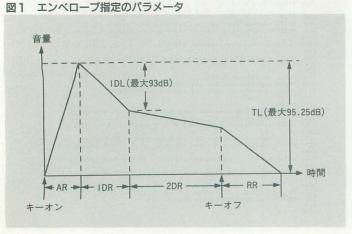
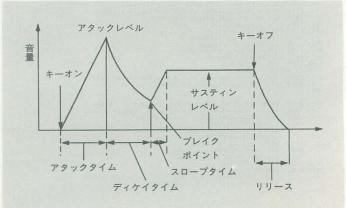
最後の中括弧の中身は当然のことながら，波形メモリに登録される波形データそのものを表しています。これは65536個までならいくつ指定してもかまいません。

このような処理が実現されたことにより，アタック部分とループ部分を分離したエフェクトを指定できるなど，各種の効果を生み出すことができるわけですが，もっとも多用されるのは音量値に用いる場合でしょう。

音量の時間的変化をエンベロープといいます。エンベロープは音色の構成上，もっとも重要な要素とされてきました。楽器音の基本波形よりもその楽器の特徴を表しているといわれています。たとえば，ピアノの基本波形とトランペットのエンベロープを合成して聞かされると，たいていの人は基本波形がどの楽器から取られたかなどは識別できません。

FM音源で指定できるエンベロープパターンは図1のようなものです。これは源音信号と変調器のすべてに指定できるので，単に音量だけでなく周波数成分の経時的変

図1　エンベロープ指定のパラメータ

図2　コルグM1のエンベロープパラメータ

化なども まとめて設定されます。本格的な音源になるともう少し込み入ってきます。参考までに図2はコルグM1のVDA1(音量の時間的変化値)を表したものです。

エンベロープは大きく,

アタック

ディケイ

サスティン

リリース

の4つの部分に分けられます。

アタックは音の出始めの変化のことで,もっとも楽器の特徴を顕著に表す部分です。

ディケイとサスティンは持続時に見せる変化のことです。FM音源ではアタックからの立ち下がりを表す1stディケイと持続と減衰を示す2ndディケイに分かれています。

リリースはキーオフ後の発音です。通常の楽器は鍵盤が離されても音は止まりません。完全に音を止めるまでの時間と考えてください。

FM音源では簡単なパターンしか使用できませんでしたが,波形メモリを用いることで,非常にリアルなエンベロープを作ることもできるようになったのです。

なお,登録されたデータはデフォルト状態では絶対音長6経過するごとに出力されます(絶対音長1は全音符の192分の1の時間)。この読み出し速度はSコマンドで設定され,次のデータを読み出すまでどの程度の間隔を置くかを指定できます。

たとえば,

S2,16

という指定ならば,ピッチモジュレーション部分では絶対音長2ごと,アンプリチュードモジュレーション部分では絶対音長16ごとにひとつ値を読み出して出力するようになります。

ですから,ひとつの波形データでも読み出し速度を変えることで違ったエンベロープにすることもできるわけです。

*　　*　　*

このように通常は波形メモリでの音量変化は音色設定のエンベロープと同様な動作になるのですが(発音と同時にデータの先頭から出力される),特殊な状況下では音色のアタックと同期せずに一連のシーケンスに対しての変化をつけたいということがあります。すなわちクレシェンドやデクレシェンドなどや,曲の流れのなかでの連続的な盛り上がりなどを表したいときです。

このような状況に対処するため,モジュレーションには非同期モードというものも用意されています。MMLでいえばHコマンドです。0で同期,1で非同期に設定され,たとえば,

H0,1

のように設定すれば,ピッチモジュレーション(またはMIDIのARCC)は同期モード,アンプリチュードモジュレーションは非同期モードに設定されます。

もっとも簡単なサンプル

リスト1を見てください。FM音源を使ったサンプルデータです。波形はかなり無茶苦茶なものが入っていますが動作には問題ありません。

FM音源でもアンプリチュードモジュレーションとピッチモジュレーションの場合ではまた変わってきます。

なお,X-BASICで使うときは,波形の設定をm_assign()の前でやらないとうまく動かないというバグがありますので注意してください。

波形番号8番以降にユーザー波形を設定し,通常のモジュレーションやARCCの処理指定に加え,Sコマンド,@Sコマンドで波形指定と読み出し速度指定を行います。

マニュアルでは,ユーザー波形の要素として指定できる数値の幅は−32768〜32767となっていますが,これは文法上指定できる限界値を表しているものであり,実際に指定すべき値の範囲とは異なることがあります。

たとえば,FM音源のアンプリチュードモジュレーションの場合は音量値をいじるものですので(最終出力は0〜127),ゆらぎとして与えるべき数値はどんなに多く見積もっても-127〜127の範囲になります。

さらにFM音源の音量設定は対数軸上に取られているので,実際に使用できる部分は@Vでいうところの60〜127くらいの範囲にすぎません。それを踏まえて値を設定していかなければなりません。表1にまとめてあるように,実際に使えそうな値の範

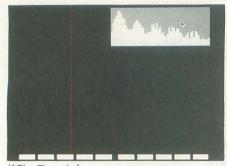
リスト1-A

```
/ FM音源のアンプリチュードモジュレーションに適用

(i)
(m1,1000)
(a1,1)

.wave_form 8, 1, 11 {-10,-5,0,5,15,20,0,-5,-8,5,10,5,15,10}

(t1) V11@200L1                        /初期設定
(t1)              cge<c>g             /通常演奏
(t1) @a1 @s,6s,8 cge<c>g              /ｱﾝﾌﾟﾘﾁｭｰﾄﾞﾓｼﾞｭﾚｰｼｮﾝ
(t1) @a1 @s,3s,8 cge<c>g              /ｱﾝﾌﾟﾘﾁｭｰﾄﾞﾓｼﾞｭﾚｰｼｮﾝ 短周期

(p)
```

リスト1-B

```
(i)
(m1,1000)
(a1,1)

.wave_form 8, 1, 11 {-10,-5,0,5,15,20,0,-5,-8,5,10,5,15,10}

(t1) V11@200L1                        /初期設定
(t1)              cge<c>g             /通常演奏
(t1) @m1 @s6 s8   cge<c>g             /ピッチモジュレーション
(t1) @a1 @s3 s,8 cge<c>g              /ピッチモジュレーション短周期

(p)
```

リスト1-C

```
(i)
(m1,1000)
(aMIDI1,1)

.wave_form 8, 1, 11 {-10,-5,0,5,15,20,0,-5,-8,5,10,5,15,10}

(t1) V11@20L1                         /初期設定
(t1)              cge<c>g             /通常演奏
(t1) @C1,0,0                          /ｺﾝﾄﾛｰﾙﾁｪﾝｼﾞ 登録
                                      /ﾓｼﾞｭﾚｰｼｮﾝﾃﾞﾌﾟｽ
(t1) @a1 @s,6s,8 cge<c>g              /

(p)
```

波形エディット中……

囲はかなり狭くなります。

実用上は±30もあれば十分でしょうか。

波形エディタ

ではさっそくこの機能を使ってみようとすると，どうやったらうまく波形を作ることができるのかという問題が浮かんできます。

直接数値入力をすればどんな波形も作れますが，とてもではありませんが感覚的な操作は期待できません。

ということで簡単な波形エディタを作成してみましょう。リスト2を見てください。

波形データは1個当たり65536個のデータを羅列できますが，これは128Kバイトの容量になります。実際に使用することを考えるとこれではあまりに大きすぎますね。ZMSファイル中に記述することを考えると64個くらいが妥当でしょうか。ここでは64ポイントの波形メモリを前提とした処理を行っていきます。

起動するとすべて0に指定された波形メモリが表示されます。マウスで棒グラフ部分を左クリックしていくと対応する部分が次々と変わっていきます。これはほぼリアルタイムで音色に反映されていきます。右クリックでZMSコマンドとしてファイルに出力できます。処理の簡略化のため，波形番号などのデータは固定されていますが，テキストエディタで取り込んで適当にエディットしてから音楽データに組み込んで使用してください。

もっと多機能にしてもよいのですが，できるだけ簡潔なものを示したかったので最低限の機能にしぼりました。これだけでも必要十分な処理は行われているはずです。

プログラム的な注意点としては，波形の書き換えの部分が挙げられます。エディット内容がリアルタイムに反映するようにしているのですが，これはMUSICZ.FNCの波形登録コマンドではX-BASICで確保された配列がそのままZ-MUSICの波形メモリとして使用されているので，うまく配列を書き換えてやれば波形の再登録をしなくても波形メモリの内容を変更できるのです。

波形登録コマンドを使うと，もともとあった配列の内容は破壊されてしまいます。配列は半分に圧縮されていますので注意してください。

波形の抽出

このように「マウスでエディットできれば……」というのは誰しも考えるところなのですが，実際に扱ってみてもそれほど絶大な効果を発揮するわけではありません(作り方次第でしょうが)。試行錯誤ができるぶん作業が楽になるのは確かですが，根本的な部分では大差はないといえます。たいていの場合リアルなエンベロープを作りたいと考えているわけですが，どんなものがリアルなのかというのがわからないのです。

となると，当然考えられるのが，実際の楽器音から波形を抽出して使うということでしょう。

ここではZ-MUSICシステムに付属するAD PCMデータのなかからSTR.PCMのエンベロープパターンを使ってみることにしました(ADDITIONディレクトリに入っ

表1 絶対音量（@V）と音量（V）の関係

Vn	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
@Vn	85	87	90	93	95	98	101	103	106	109	111	114	117	119	122	125	127

リスト2

```
 10 int a,b,c,d,e,f,g,h,x,y,r,l
 20 char v(4,10)
 30 v={31,0,0,0,0,29,0,1,0,0,0,
 40    31,0,0,0,0,24,0,3,0,0,0,
 50    31,0,0,0,0,25,0,0,0,0,0,
 60    31,0,0,0,0,8,0,3,0,0,0,
 70    4,4,15,3,0,0,0,0,0,0,0}
 80 int p(64),q(63)
 90 screen 1,3,1,1
100 fill(0,0,511,511,6353)
110 fill(245,10,500,138,35321)
120 line(246,74,500,74,45472)
130 for i=0 to 63
140   fill(246+i*4,74,246+i*4+2,136,9999)
150 next
160 mset()
170 mouse(1):mouse(4)
180 repeat
190 msstat(x,y,l,r)
200 if l<>0 then {
210   mspos(x,y)
220   if point(x,y)<6354 then continue
230   a=(x-245)/4 :c=a mod 2
240   b=74-y
250   if p(a)<b then fill(246+a*4,74-p(a),246+a*4+2,74-b,9999)
260   if p(a)>b then fill(246+a*4,74-p(a),246+a*4+2,74-b,35321)
270   p(a)=b
280   if c=1 then q(a/2)=(p(a-1) shl 16)+b else q(a/2)=p(a+1)+(b shl 16)
290 }
300 if r=-1 then zmsout()
310 until 0
320 end
330 func mset()
340 m_init()
350 m_wave_form(8,1,0,q)
360 m_fmvset(1,v)
370 m_alloc(1,5000)
380 m_assign(1,1)
390 m_trk(1,"t80l1v10@1@s,3s,8@a1[do]c&c&CC&Cefg<c>[loop]")
400 m_play()
410 endfunc
420 func zmsout()
430 str f,cr
440 int a
450 cr= chr$(13)+chr$(10)
460 input"ファイル名";f
470 a=fopen(f,"c")
480 fwrites(".wave_form 8,1,0 {"+cr+"    ",a)
490 for i=0 to 7
500   for j=0 to 7
510     fwrites(str$(p(i*8+j))+",",a)
520   next
530   fwrites(str$(p(i*8+j))+cr+"    ",a)
540 next
550 fwrites(" }"+cr,a)
560 fclose(a)
570 endfunc
```

ています。ちなみにver. 1にも同じものが収録されています)。

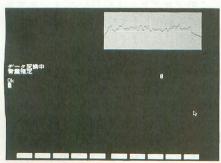
周波数成分はほとんど意味がありませんのでパワースペクトルだけで見ていきます。ここで描かれたグラフの高低は音圧の差が表れているということになります。これはほぼ音量と見てもかまわないでしょう。

自然音の場合，振幅がそのままエネルギーを表していますから振幅を取ってみます。波形からピークレベルだけを使ってエンベロープを抽出します。具体的には波形を小区間に分割して，その中での最大値と最小値を拾っていきます。その差を音量とみなしてみることにしましょう。

まず，AD PCM→PCM変換が必要になります。これはMUSICZ.FNCでサポートされていますのでそのまま使いましょう。ADPCM_TO_PCM()ではCHAR型の配列の中身をINTの配列に変換します。元のCHAR型配列には配列の1要素に2個分のAD PCMデータが詰まっていましたが，変換後は16ビットPCMデータ1個でINT型配列の1要素（32ビット）を占有します。

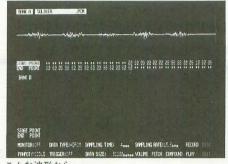
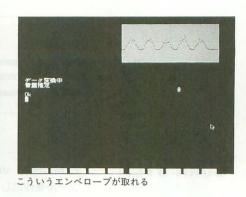
このスカスカのデータも，波形メモリ登録時には「与えられた配列そのもの」が充填構造に書き換えられてしまうので注意が必要です。

このプログラムでは波形抽出のテストだけを行っており，ZMSコマンドへの出力などは省略されていますが，リスト2を参考にすればこの部分はすぐに拡張できるでしょう。

なお，ここではAD PCMデータの波形から音量変化を抽出したわけですが，これではアンプリチュード（音量）モジュレーションには使えても，ピッチ（音程）モジュレーションには適当でないともいえます。波形から音程情報を抽出するにはゼロクロス点をカウントするなどの方法があるので（一定時間内に何度データの符号が変わるかを調べる），興味のある方は試してみるのも面白いかもしれません（この場合一度ローパスフィルタを通す処理を加えたほうがよいでしょうが）。

STR.PCMを抽出する

こんな波形から……

こういうエンベロープが取れる

＊　　＊　　＊

波形メモリによって音楽表現の可能性が大きく広がったのがわかると思います。今回はアンプリチュードモジュレーションを中心に扱いましたが，MIDIのARCCに適用すればその効果は計り知れないものがあります。こうしてこれまで機械的になりがちだったMMLの音楽データも，これでよりリアルなものになっていくと思います。

せっかくの新機能ですから皆さんもぜひ活用してみてください。

リスト3

```
 10 int a,b,c,d,e,f,g,h
 20 char v(4,10)
 30 v={31,0,0,0,0,29,0,1,0,0,0,
 40    31,0,0,0,0,24,0,3,0,0,0,
 50    31,0,0,0,0,25,0,0,0,0,0,
 60    31,0,0,0,0,8,0,3,0,0,0,
 70    4,4,15,3,0,0,0,0,0,0,0}
 80 str l[255],m[255]
 90 int p(255),q(63)
100 int x(255),y(255)
110 char ad(20000)
120 int pc(40000)
130 screen 1,3,1,1
140 fill(0,0,511,511,6352)
150 fill(246,10,501,138,35321)
160 line(246,74,501,74,45472)
170 a=fopen("str.pcm","r")
180 b=fseek(a,0,2)
190 fseek(a,0,0)
200 fread(ad,b,a)
210 adpcm_to_pcm(ad,b,pc)
220 c=b/256
230 locate 0,12
240 print "データ変換中"
250 for i=0 to b*2-1
260   d=pc(i)
270   if d>32767 then pc(i)=d-65536
280 next
290 print "音量推定"
300 for i=0 to 255 :d=-1000:f=1000
310 locate 48,14:print 255-i;
320  for j=0 to c-1
330   e=pc(i*c*2+j*2)
340   if d<e then d=e
350   if f>e then f=e
360  next
370  p(i)=(d-f)/40-20
380 next
390 wavedraw()
400 for i=0 to 63
410  q(i)=p(i*4)
420 next
430 mset()
440 end
450 func mset()
460 m_init()
470 m_wave_form(8,1,0,q)
480 m_fmvset(1,v)
490 m_alloc(1,5000)
500 m_assign(1,1)
510 m_trk(1,"t80l1v10@1@s,3s,8@a1c&c&CC&C&Cefg<c>")
520 m_play()
530 endfunc
540 func wavedraw()
550 for i=0 to 255
560 pset(i+246,74-p(i),8)
570 next
580 endfunc
```

リアルタイム楽譜表示を目指す

X-BASICで音楽ツールを作る

Morishita Yasuyuki 森下 泰行

COMPUTER MUSIC

Z-MUSIC ver.2.0とMUSICZ.FNCを使ってX-BASICでサポートツールを作成してみました。演奏されている内容からMMLを割り出し，リアルタイムに楽譜表示します。

Z-MUSICのマニュアルには載っていませんが，Z-MUSICシステムの専用外部関数MUSICZ.FNCにはZ-MUSICの演奏状態を覗き見るための関数ZM_WORK()というものがあります。

これは現在演奏中のデータの情報を見るもので，たとえば，トラック1で演奏しているのはどんな楽器かとか，音程はどうなのか，モジュレーションがどの程度かかっているのかといったことを簡単に調べることができます。

使い方は，

ZM_WORK(TRACK,OFFSET)

のようにし，そのトラックの指定されたオフセットにあるワークエリアの値をCHAR型で返してきます。1バイトずつ読みますのでロングワードデータなどの場合は組み合わせて使わなければなりません。

たとえば，トラック3の音量を見たいときは，

ZM_WORK(3,&H1F)

となります（トラックワークについての詳細はZ-MUSICのマニュアルをご覧ください）。

トラックワークの知識が必要なのですが，その気になれば，鍵盤表示プログラムやレベルメーターなどを持った支援ツールをX-BASICで記述することができるでしょう。

ミュージックビュア

ところで，X68000のフリーソフトウェアでは，まさに“ビュア”と呼べるようなさまざまなミュージックプレイヤーが発表されています。

単純に表示を楽しむもの，開発支援を目的としたものに分かれますが，表示される内容はどれも大差ないものになっています。もう少しユニークなものが出てきてもよさそうな気がするのですが。

ゲーム機3DO REALでは音楽CDを入れると演奏内容にあわせて画面上の模様が変化していくという機能がありました。雑誌の紹介記事では音楽の種類によって表現が変わるようなことが書いてあったので非常に興味を持っていたのですが，実際に見てみるとどんなジャンルの曲でもほとんど差は出てきませんでした。しかし，音楽のビジュアライズに対する姿勢は高く評価してもいいでしょう。

X68000にももっと遊び心を持ったミュージックビュアがほしいところですが，どんな情報をどのように可視化するべきかというのはまだまだノウハウが必要な部分でもあります。で，試行錯誤にはやはりX-BASICから使える関数が便利です。

多くのツールでは鍵盤表示になっていますが，それだけでは現在MMLのどこを演奏しているのかを知ることができません。時系列に沿った情報を表示させるものがひとつくらいはほしいところです。

ここでは，いずれ期待されるであろうビジュアライズの一環としてMML表示と楽譜表示を行ってみました。

基本方針

Z-MUSICの機能は膨大です。表示に必要ないものもたくさんありますし，ループ命令などを細かく追っていくのは非常につらい作業となるでしょう。絶対音長が使え，トラック間の同期も不要ですし，トラックごとに独立したループが組めますから，通常の楽譜表記では再現できないようなデータも堂々とまかりとおっています。

ですから，元のMMLソースファイルがあっても楽譜データに変換することは簡単ではありません。以前，MMLデータをMUSIC PRO-68Kの楽譜データに変換するMML2SCOが掲載されましたが，これはツール側が時間管理を行い各トラックの非同期ループをすべて展開して再整理するという方法をとっていました。かなり面倒ですが，おそらくほかに有効な方法はないでしょう。

演奏されているデータのワークエリアから解析してやれば，Z-MUSICの複雑な制御構造も回避できますし，ZMS以外のデータでも演奏されていればそのまま楽譜が表示できます。音長判定が若干甘くなりますが，ある程度はいたしかたありません。

ということで，このような技術を確立しておくこともあながち無駄ではないといえるでしょう。

この方法では原理的にいっても実際のソースMMLと同じものを再現することはできません。また，8トラック分も表示エリアをとってしまえば1トラック当たりの表示エリアは2，3行しかありません。このエリア内ですべての情報を書いていたのではスクロールしっぱなしでとてもテキストを読むことはできなくなるでしょう。ということで，テキストとして表示する部分には最低限の情報だけを記述したほうがよいことになります。ここでは音階と音長，音色切り換えのみを表示することにしました。また，データによっては休符が邪魔になることがわかりましたのであえて表示していません。

入力と使い方

このプログラムはX-BASICインタプリタ上でも動作しますが，音長計算の精度は無茶苦茶になりますので，コンパイルして使用したほうがいいでしょう。

なお,XCではコンパイルできないかもしれません。少なくとも私の環境ではできませんでした。できるだけBC+GCCを使ってください。GCCで2進数表現を通さないバージョンをお使いの場合は，定数値を十進数で指定しなおしてください。

また，スプライト定義部分はあまりに大

きくなってしまったのでリスト2に圧縮してあります。データサイズ1551バイトでMAC.Xを使って入力し，LHAで展開したものをプログラムに加えてください。

使い方ですが，Z-MUSICでなんらかの曲を演奏しているときにこのツールを立ち上げてください。曲を再演奏してMMLと楽譜を表示します。MMLは連続した8トラック分，楽譜は3トラック分の音程表示，そのうちの最初のトラック分の楽譜表示を行います。

キー操作はROLL UP/DOWNでMML表示トラックの切り換え，テンキーの7，1で1現象めのトラック指定，8，2で2現象め，9，3で3現象めの指定トラックを切り換えます。

プログラム解説

Z-MUSICでは1トラックで8音，最大で640音の発音を管理できますが，とても表示しきれませんのでこのプログラムでは最初の1音しか見ていません。

参照しているのはZ-MUSICのトラックワークのうち，

ステップタイム	&H01
ゲートタイム	&H03
ノートナンバー	&H42
タイ	&HBD
音色番号	&H1D

くらいのものです。

なお，ステップタイムとは次の命令を処理するまでの待ち時間を表し，ゲートタイムとは実際の発音時間を表します。音長はステップタイムをもとにして算出し，ゲートタイムカウンタはスクロールの基本クロック代わりに使っています。

音長計算は力技です。指定されたステップタイムをできるだけ自然な音楽的音長に変換するようにしてみました。どうしても該当するものがないときは絶対音長で表示します。

X-BASICの中級者がこのプログラムの動作を見ると「おや」っと思う部分があるかもしれません。X-BASICには制限がたくさんあります。一見して面倒そうなスプライトパターンは愛と勇気があれば作れます。派手な画面もX68000のハード構成を熟知していればなんとかなりそうです。ではなにが問題かというとトラックMML表示エリアの逆スクロール処理部分です（テキスト画面）。

それぞれのトラック情報はコンソールを切ったウィンドウ内に表示されていますので単にPRINTするだけでかまいません。それらはROLL UP/DOWNキーを操作することによって上下にスクロールします。

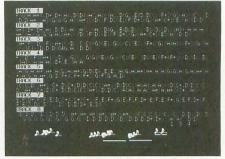
要はそれぞれのカーソル位置を保存して画面を部分スクロールさせればいいのですが，X-BASICの通常命令ではテキスト画面の逆スクロールを行うことはできません。もちろんいちいち再描画していてはツールの精度に関わってきます。

そこでここではDOSのエスケープシーケンスを使って画面を逆スクロールさせています。これらはX-BASICのPRINT命令では発効しませんので，ファイル“CON”に対して直接書き込みを行います。リストを見てファイル処理部分に疑問を感じる向きもあるでしょうが，とりあえずこのままで動作します。

はっきりいって裏技ですので積極的に活用をおすすめはしませんが，DOS上で動くBASICというのを実感するにはよい題材かもしれません。

技術的難点

表示にスプライトを用いているので画面の横幅を512ドット以上に上げることができません。細かい表示を行うにも限界があります。また，水平表示32個というのもけっこう厳しい壁になってきます。

実際には同じラインに32個以上並ぶことはそれほどないので表示限界をもっと多く設定していてもあまり問題はありません。しかし，露骨に並ぶときもありますので一応32個までの表示にとどめておきました。

リアルタイムに音符を表示

それに比べればスプライト表示限界の1画面128個というのはあってなきがごとき限界です。適当なライブラリさえ用意すればいくらでも拡大できますので無視してもなんの問題もないでしょう。

ありがちなことに，このプログラムも「いつかは楽譜エディタ……」を目指して作成されたものです。肝心のエディット部分はややこしそうですし，仕様も固まっていませんので実現するかどうかはまだわかりません。

しかし実際になんの役に立つかと聞かれると，ちょっと難しいところがあります。開発用というよりは解析用，学習用のものだと考えています。エディット時と演奏時にリアルタイムで楽譜を動かすための基礎技術だと思えばよいでしょう。

プログラムとしては，表示を通常関数のみで行っているので表現には限界があるということ，楽譜部分の多トラック化が課題といえるでしょう。スプライトの使い方にまだまだ無駄がありますし，スプライトダブラー系の技術を使えば表示内容をもっと充実させることができるようになるはずです。もう少し強力な表示用外部関数（テキストでのマルチフォントなど）がほしいところで……。

リスト1

```
 1: str p,l,dam,oc
 2: int b,d,kc,k2,t,o,nn,tp=1,tf,tpp,da
 3: int oo(79),a(79),x(79),y(79),n(79)
 4: int gsc,gx,gy,gkc,gb,m,gd2=5,gd3=3,rl,rr,spp
 5: int yd(511,2,7),rf(79),rs(79)
 6: int sps(50),spi,sx,sff
 7: int spx(50),spy(50)
 8: int gc(2),gd=1,gx1(2),gy1(2),gb1(2),gs,gq,u
 9: char too(623),he(255)
10: /* ------------------------- 初期画面を作る-----------
11: screen 1,2,1,1
12: apage(1)
13: sp_init()
14: sp_on():sp_disp(1)
15: sprite_pattern()
16: sprite_pattern2()
17: for i=0 to 2:gc(i)=400:gbl(i)=500:next
18: color[0,&B111001110101111,&B110110111001101,&B1001110011100110]
19: fill(0,0,511,511,1)
20: fill(0,388,511,511,6)
21: gp()
22: locate 50,24:print gd;gd2;gd3
23: for i=0 to 8:line(0,i*48,511,i*48,2):next
24: for i=0 to 8:line(0,i*48+1,511,i*48+1,3):next
25: palet(1,&B11000110001101)
26: palet(2,&B1010010100l0100)
27: palet(3,&B1000010000101)
28: palet(4,&B1110011100111101)
29: palet(5,&B101001010011101)
30: palet(6,&B1000100001101)
31: for i=1 to 5 :line(0,450-i*4,511,450-i*4,4):next
32: for i=6 to 16:line(0,450-i*4,511,450-i*4,5):next
33: line(0,450,511,450,5)
34: for i=-5 to -1:line(0,450-i*4,511,450-i*4,4):next
35: for i=-16 to -6:line(0,450-i*4,511,450-i*4,5):next
36: apage(0)
37: console 0,32,0
38: fopen("con","w")
39: for i=0 to 7
40:  locate 0,i*3 :color 14:print "TRACK ";i+1:color 3
41: next
42:  m_play()
43: for i=0 to 8
44:  y(i)=i*3+1
```

```
 45: next
 46: for i=10 to 79
 47:  y(i)=25
 48: next
 49: /*++++++++++++++++++++++++++++++++++++ main loop ========*
 50: repeat
 51:  console 0,24,0
 52:  tf=0
 53:  da=asc(inkey$(0))
 54:   if da=15 then {
 55:    if tp<>1 then {
 56:     tf=-1:y(tp-1)=-2
 57:     locate 0,0:fputc(27,0):fputc('M',0)
 58:     fputc(27,0):fputc('M',0):fputc(27,0):fputc('M',0)
 59:     locate 0,0:color 14:print"TRACK ";tp-1;
 60:     color 3:x(tp-1)=0
 61:    }
 62:   }
 63:   if da=14 then {
 64:    if tp<>71 then {
 65:     tf=1 :y(tp+8)=25
 66:     locate 0,23:print:print:print
 67:     locate 0,21:color 14:print "TRACK ";tp+8;
 68:     color 3:x(tp+8)=0
 69:    }
 70:   }
 71:  console 0,32,0
 72:  locate 50,24
 73:  if da='7' then if gd>0   then gd=gd-1  :print gd;gd2;gd3;
 74:  if da='1' then if gd<79  then gd=gd+1  :print gd;gd2;gd3;
 75:  if da='8' then if gd2>1  then gd2=gd2-1:print gd;gd2;gd3;
 76:  if da='2' then if gd2<79 then gd2=gd2+1:print gd;gd2;gd3;
 77:  if da='9' then if gd3>0  then gd3=gd3-1:print gd;gd2;gd3;
 78:  if da='3' then if gd3<79 then gd3=gd3+1:print gd;gd2;gd3;
 79:  tp=tp+tf
 80:  for i=tp to tp+7
 81:   disp(i)
 82:  next
 83:  if scrol()=1 then {
 84:   sx=(sx+1) mod 480
 85:   for i=0 to 31
 86:    if ((spx(i)-sx)>40)and(spy(i)>200) then {
 87:     sp_move(i,spx(i)-sx,spy(i),sps(i))
 88:    } else {
 89:     sp_move(i,0,-16,255):spy(i)=-16
 90:    }
 91:   next
 92:   scr(gd,0)
 93:   if sff=1 then { spy(spi)=gy-13:spx(spi)=507
 94:    sp_move(spi,507,spy(spi),sps(spi)):sff=0
 95:   }
 96:   scr(gd2,1)
 97:   scr(gd3,2)
 98:   if spi>30 then spi=0
 99:  }
100: until da=27
101: /*+++++++++++++++++++++++++++++++++++++++++++++++++++=*
102: console 0,31,1
103: end
104: /*    以下，各種外部関数
105: /*
106: /*     -------------------- 怒涛の解析表示関数 ------------
107: func disp(d)
108: int ei=0
109: console (d-tp)*3+1,2,0
110: locate x(d),y(d)-tf*3
111: spp=100
112: nn=zm_work(d,&H1D)
113: b=zm_work(d,1)
114: if n(d)<>nn then color 2 :print "@";str$(nn); :color 3 :n(d)=nn
115: if a(d)<b then {
116:  kc=zm_work(d,&H42)
117:  o=kc /12
118:  if o<>oo(d) then {
119:   switch oo(d)-o
120:    case  1 :oc=">":break
121:    case -1 :oc="<":break
122:    default :if o<9 then oc="O"+str$(o-1)
123:   endswitch
124:   if o<9 then  oo(d)=o
125:  } else oc=""
126:  if o>8 then p="R" else {
127:   switch kc mod 12
128:    case 0:  p="C"   :break
129:    case 1:  p="C+":ei=2  :break
130:    case 2:  p="D"   :break
131:    case 3:  p="D+":ei=2  :break
132:    case 4:  p="E"   :break
133:    case 5:  p="F"   :break
134:    case 6:  p="F+":ei=2  :break
135:    case 7:  p="G"   :break
136:    case 8:  p="G+":ei=2  :break
137:    case 9:  p="A"   :break
138:    case 10: p="A+":ei=2  :break
139:    case 11: p="B"
140:   endswitch
141: /*               ----------3連符関係の処理 --------------
142:   if (b+1)/2=4 then { spp=88:if rs(d)<>8 then { rs(d)=8
143:                     p="{"+p  :rf(d)=1  } else rf(d)=rf(d)+1
144:    if rf(d)=3 then p=p+"}":rf(d)=0:rs(d)=0:l="8" else l=""
145:   }
146:   if (b+1)/2=8 then { spp=92:if rs(d)<>4 then { rs(d)=4
147:                     p="{"+p  :rf(d)=1  } else rf(d)=rf(d)+1
148:    if rf(d)=3 then p=p+"}":rf(d)=0:rs(d)=0:l="4" else l=""
149:   }
150:   if (b+1)/2=16 then { spp=96:if rs(d)<>2 then { rs(d)=2
151:                     p="{"+p  :rf(d)=1  } else rf(d)=rf(d)+1
152:    if rf(d)=3 then p=p+"}":rf(d)=0:rs(d)=0:l="2" else l=""
153:   }
154:   if not (((b+3)/4=2)or((b+3)/4=4)or((b+3)/4=8)) then {
155:    if rf(d)=1 then {
156:     if rs(d)=2 then print "}";:color 1:print"*32";:color 3
157:     if rs(d)=8 then print "}";:color 1:print"*8" ;:color 3
158:     if rs(d)=4 then print "}";:color 1:print"*16";:color 3
159:    }
160:    if rf(d)=2 then {
161:     if rs(d)=8 then print "}";:color 1:print"*16";:color 3
162:     if rs(d)=4 then print "}";:color 1:print"*32";:color 3
163:     if rs(d)=2 then print "}";:color 1:print"*64";:color 3
164:    }
165: /*               ----------------- 音長解析 -------------
166:    switch (b+3)/4
167:     case 12:  l="4":spp=40:break
168:     case 24:  l="2":spp=56:break
169:     case  6:  l="8":spp=24:break
170:     case  3:  l="16":spp=12:break
171:     case 48:  l="1":spp=72:break
172:     case 18:  l="4.":spp=44:break
173:     case  9:  l="8.":spp=28:break
174:     case 72:  l="1.":spp=76:break
175:     case 36:  l="2.":spp=60:break
176:     case 21:  l="4..":spp=48:break
177:     case 60:  l="1^4":break
178:     case 42:  l="2..":spp=64:break
179:     case 54:  l="1^8":break
180:     case 45:  l="2...":spp=68:break
181:     case 30:  l="2^8":break
182:     case 51:  l="1^16":break
183:     case 15:  l="4^16":break
184:     case 27:  l="2^16":break
185:     case 39:  l="2.^16":break
186:     default:  l="*"+str$(b)
187:    endswitch
188:    switch (b+1)/2
189:     case  3:  l="32":spp=4:break
190:     case  9:  l="16.":spp=16:break
191:     case 15:  l="8^32":break
192:     case 51:  l="2^32":break
193:     case 21:  l="8..":spp=32:break
194:     case 27:  l="4^32":break
195:     case 45:  l="4...":spp=52:break
196:    endswitch
197:    if b=3  then l="64":spp=0
198:    if b=9  then l="32.":spp=8
199:    if b=15 then l="16^64"
200:    if b=27 then l="8^64"
201:    if b=21 then l="16..":spp=20
202:    if b=45 then l="8...":spp=36
203:    rs(d)=0:rf(d)=0
204:   }
205:   print oc;p;
206:   color 1:print l;
207:   if zm_work(d,&HBD)=255 then print"&";:spp=spp+1
208:   color 3
209:  }
210:  if (d=gd) and (o<9) then { sff=1 :spi=spi+1
211:   for i=0 to 31
212:     spx(i)=spx(i)-sx
213:   next
214:   sx=0:sps(spi)=spp+ei
215:  }
216: }
217: a(d)=b
218: x(d)=pos:y(d)=csrlin
219: endfunc
220: /*-----------------------------------------------------
221: func scrol()
222: gb=zm_work(gd,1)
223: rl=0
224: if gbl(0)<>gb then  {
225:  gsc=(gs+513)mod 512
226:  home(0,gsc,0)    :rl=1
227:  gbl(0)=gb:gs=gs+1
228: } else {
229:  gb=zm_work(gd2,1)
230:  if gbl(1)<>gb then  {
231:   gsc=(gs+513)mod 512
232:   home(0,gsc,0)  :rl=1
233:   gbl(1)=gb:gs=gs+1
234:  } else {
235:   gb=zm_work(gd3,1)
236:   if gbl(2)<>gb then  {
237:    gsc=(gs+513)mod 512
238:    home(0,gsc,0) :rl=1
239:    gbl(2)=gb:gs=gs+1
240:   }
241:  }
242: }
243: return(rl)
244: endfunc
245: /*-----------------------------------------------------
246: func scr(gd,u)
247: gq=zm_work(gd,3)
248: gkc=zm_work(gd,&H42)
249: if gkc<127 then {
250:  gx=(gsc+510) mod 512
251:  o=(gkc¥12)*7
252:  switch gkc mod 12
253:   case  0: m=0:e=0 :break
254:   case  1: m=0:e=1 :break
255:   case  2: m=1:e=0 :break
256:   case  3: m=1:e=1 :break
257:   case  4: m=2:e=0 :break
258:   case  5: m=3:e=0 :break
259:   case  6: m=3:e=1 :break
260:   case  7: m=4:e=0 :break
261:   case  8: m=4:e=1 :break
262:   case  9: m=5:e=0 :break
263:   case 10: m=5:e=1 :break
264:   case 11: m=6:e=0
265:  endswitch
266:  gy=520-(o+m)*2
267:  if rl<5 and gy>300 then {
268:   pset(gx,gy,255):pset(gx1(u),gy1(u),55+e+u*92):gc(u)=gc(u)+1
269:   yd(gx,u,rl)=gy:rl=rl+1
270:   gx1(u)=gx:gy1(u)=gy
271:  }
272: }
273:  for i=0 to 4
274:   pset((gsc+50) mod 512,yd((gsc+50)mod 512,u,i),0)
275:  next
276: endfunc
277: /*
278: func gp()
279: too={
280: 6,6,6,6,6,6,4,6,6,6,6,6,6,
281: 6,6,6,6,6,4,4,4,6,6,6,6,6,
282: 6,6,6,6,4,4,4,4,6,6,6,6,6,
283: 6,6,6,6,4,4,6,4,4,6,6,6,6,
284: 6,6,6,6,4,4,6,6,4,6,6,6,6,
```

▶MT-32を1万円で買った(中古)。RS-232Cとは導線と抵抗だけで繋いだのでケーブル代はタダ。しかし，説明書がない。　　吉田　晃(18)東京都

```
285: 6,6,6,6,4,4,6,0,4,6,6,6,6,
286: 6,6,6,6,6,4,6,4,4,6,6,6,6,
287: 6,6,6,6,6,4,4,4,4,6,6,6,6,
288: 6,6,6,6,6,4,4,4,6,6,6,6,6,
289: 6,6,6,6,4,4,4,4,6,6,6,6,6,
290: 6,6,6,4,4,4,4,6,6,6,6,6,6,
291: 6,6,4,4,4,6,4,6,6,6,6,6,6,
292: 6,4,4,4,4,6,4,4,4,4,6,6,6,
293: 6,4,4,4,6,4,4,4,4,4,4,4,6,
294: 4,4,4,6,4,4,4,4,6,4,4,4,6,
295: 4,4,4,6,4,4,6,4,6,6,4,4,4,
296: 4,4,4,6,4,4,6,4,6,6,4,4,4,
297: 4,4,4,6,6,4,4,4,4,6,4,4,4,
298: 4,4,4,6,6,6,6,6,4,6,4,4,4,
299: 6,4,4,6,6,6,6,6,4,6,4,4,6,
300: 6,6,4,4,6,6,6,6,4,4,4,6,6,
301: 6,6,6,6,4,4,4,4,4,4,6,6,6,
302: 6,6,6,6,6,6,6,6,6,4,6,6,6,
303: 6,6,6,6,6,4,4,6,6,4,6,6,6,
304: 6,6,6,6,6,4,4,6,6,4,6,6,6,
305: 6,6,6,6,6,4,6,6,6,4,6,6,6,
306: 6,6,6,6,6,6,4,4,4,6,6,6,6}
307: he={
308: 6,6,6,6,6,6,4,6,6,6,6,6,6,6,6,
309: 6,6,4,4,4,4,4,4,6,6,6,6,6,6,6,
310: 6,4,4,4,4,6,6,4,4,4,6,6,6,6,6,
311: 4,4,4,6,6,6,6,6,4,4,4,6,6,4,4,
312: 4,4,6,6,6,6,6,6,4,4,4,4,6,4,4,
313: 4,4,4,4,6,6,6,6,6,4,4,4,6,6,6,
314: 4,4,4,4,6,6,6,6,6,4,4,4,6,6,6,
315: 6,4,4,6,6,6,6,6,6,4,4,4,6,4,4,
316: 6,6,6,6,6,6,6,6,6,4,4,4,6,4,4,
317: 6,6,6,6,6,6,6,6,6,4,4,4,6,6,6,
318: 6,6,6,6,6,6,6,6,4,4,4,6,6,6,6,
319: 6,6,6,6,6,6,6,6,4,4,4,6,6,6,6,
320: 6,6,6,6,6,6,6,6,4,4,6,6,6,6,6,
321: 6,6,6,6,6,6,6,4,4,4,6,6,6,6,6,
322: 6,6,6,6,6,6,4,4,4,6,6,6,6,6,6,
323: 6,6,6,6,6,4,4,6,6,6,6,6,6,6,6,
324: 6,6,6,4,4,6,6,6,6,6,6,6,6,6,6}
325: put(15,425,27,452,too)
326: put(15,453,29,469,he)
327: endfunc
328: /*
```

以下，スプライトパターン

リスト2

```
000000  1B B4 2D 6C 68 35 2D F1 : 23
000008  05 00 00 72 1B 01 00 11 : A4
000010  1B B9 06 20 01 02 73 70 : E0
000018  4D 47 48 00 00 03 32 6A : 7B
000020  99 E1 C2 B6 D7 FB 9E 85 : E7
000028  E9 78 A7 24 C1 78 3F FB : 9F
000030  DE F7 BD EF 7B 81 F8 1B : 90
000038  4A 7A CA 6B C1 A1 49 A5 : 49
000040  E2 47 E3 73 35 33 49 6C : 9C
000048  F4 92 DB A4 9E F7 AC 8C : D2
000050  92 4B E9 3D 45 E2 78 11 : B3
000058  0C 8E 45 55 2D 05 67 91 : 5E
000060  11 E2 41 4D 87 28 3C 4F : BB
000068  2B CE BC 8F C0 24 CA 0D : FF
000070  C6 DB 39 5C EC AB 96 BC : 1F
000078  0E E0 6D 1D 90 CA 2F 3E : 3F
-------------------------------------
CKSUM:  B6 9B FA 30 60 A2 8F 0C 2090

000080  37 12 09 81 4F DC 3D EC : 27
000088  3A 59 52 57 33 3E 6B B2 : CA
000090  B8 79 D3 B3 67 CC C9 CE : 81
000098  CA 9F 3E 64 E9 B2 F8 18 : B6
0000A0  F8 D2 A5 65 E6 F0 A5 3B : 8A
0000A8  C1 CA 9D 29 D9 6A B0 C7 : 0B
0000B0  13 F1 3B DE F7 D4 3E A5 : CB
0000B8  4A 4C 8D 1F E5 18 A1 FE : DE
0000C0  4A 5F D5 B0 F4 F1 4F E2 : 44
0000C8  29 62 FC 8C 4A 47 C4 AA : 12
0000D0  62 EC 71 24 B9 88 F5 F8 : 11
0000D8  97 95 BD F4 BA 86 FE 26 : 41
0000E0  B7 5B 9E C4 D1 C3 C7 8A : 59
0000E8  4B F8 9D 4C 32 50 C3 F7 : 68
0000F0  F4 3E 38 79 D9 39 73 33 : 9B
0000F8  25 A6 43 A0 B1 05 8F B4 : A7
-------------------------------------
CKSUM:  90 D5 2B F7 AB 75 2F 3B A0AC

000100  2C 61 08 F4 B8 7D 2E 49 : 35
000108  C5 FF 3C 2E 94 D6 17 5A : 09
000110  3B B1 FE 67 47 49 D9 17 : D1
000118  BA 5D 67 B4 25 A9 EF 47 : 36
000120  F1 AF 7E D8 7B A1 4B A8 : 05
000128  F6 83 C3 DA D2 00 F7 A9 : 88
000130  AC EB 71 9A DA CD 2C 63 : D8
000138  35 35 9D CE C9 26 BF 59 : DC
000140  7F 65 8E 54 58 E1 63 BA : 1C
000148  4B 1C B0 2E BE 31 74 98 : 40
000150  05 CA 46 2E 13 EF 56 95 : 30
000158  C1 5C 0A EE 82 B3 30 4B : C5
000160  D7 72 5E A9 1C BD 13 66 : A2
000168  5E 7F 82 C4 32 0D 00 68 : CA
000170  76 56 A9 A2 0C 44 18 B2 : 31
000178  2B 19 5B C2 E6 2C 66 D2 : AB
-------------------------------------
CKSUM:  14 C7 6A C6 93 C7 28 92 143D

000180  C6 36 0B 20 59 72 85 96 : 0D
000188  25 6E 70 82 41 04 8D E5 : 3C
000190  90 51 01 AA 0D 5E D2 D9 : A2
000198  A8 A1 07 0B 3B 3F 62 5F : 96
0001A0  5B 13 24 43 CC 6B E4 AC : 9C
0001A8  71 EB A0 05 7F 47 C1 16 : 9E
0001B0  A2 C4 11 E3 82 3C 23 32 : 6D
0001B8  AF 6A AB 8D 0D 5D 68 87 : AA
0001C0  98 DF CD 58 A5 C1 A2 0D : B1
0001C8  1F DD 6B 18 20 CA 41 97 : 41
0001D0  9B E0 A8 64 16 60 B3 FA : AA
0001D8  BC 10 CD 10 4B 20 97 BE : 69
0001E0  B2 4D 83 54 1A BF C2 DA : 4B
0001E8  38 41 C4 83 8E 45 DF 4E : C0
0001F0  B5 C1 7D 6F 1D B4 B2 CA : AF
0001F8  A0 2D C1 6F EE 0B 7C 4A : BC
-------------------------------------
CKSUM:  8D EA 35 A8 95 2C 72 C6 D223

000200  F0 AA 84 15 C8 2B F6 56 : 72
000208  69 50 6C 83 67 FB 5B C5 : 2A
000210  BB 70 7D 48 A0 F9 B3 2E : 6A
000218  04 3D D7 9C 95 8D 5C 15 : 47
000220  40 AA DC F5 DD 2A C7 FF : 88
000228  04 14 8E 08 0F 87 7E 37 : F9
000230  69 CF 74 87 BC B0 0C 83 : 2E
000238  30 19 9B EB 44 D1 05 62 : 4B
000240  0A DF 2F AF 15 56 C1 5C : 4F
000248  0A EE 7F AE 79 57 08 1E : 1B
000250  10 3C EB AC 42 C8 0D 00 : FA
000258  68 76 16 A9 65 08 31 10 : 4B
000260  62 F9 FD 75 2B 2A 0B 20 : 4D
000268  59 74 BD 70 4B 2C 41 20 : D2
000270  82 46 F2 C8 2E 0D 40 6A : 67
000278  76 96 C9 82 0D C4 1B A0 : E3
-------------------------------------
CKSUM:  34 15 E1 CC 36 82 64 4D EB08

000280  11 9A 1F 71 8B D8 32 1B : EB
000288  6B 28 C8 2D 81 6D CB 16 : 57
000290  D2 2B B3 44 15 08 2A 76 : B1
000298  16 61 B0 6C 03 63 FA 5B : 4E
0002A0  A7 2C C1 E9 2F C1 E4 C3 : 14
0002A8  7C 18 6F 14 56 DC 52 5F : FA
0002B0  0F 15 B4 AF 4B A0 26 01 : 99
0002B8  31 B6 26 31 29 D2 EA 5A : 7D
0002C0  80 E2 45 01 BB 78 8A DB : 40
0002C8  13 C2 76 44 C3 23 C8 A0 : DD
0002D0  23 F9 6A 9B F8 8D F5 5C : F7
0002D8  54 17 40 BA FB 95 F9 62 : 50
0002E0  04 1D D8 10 41 E4 91 AF : 6E
0002E8  56 3F 75 58 17 04 E0 27 : 84
0002F0  3D 21 39 22 A1 30 40 18 : E2
0002F8  80 37 4D 59 19 05 F0 2F : 9A
-------------------------------------
CKSUM:  E8 C5 8C A8 A0 99 48 D5 04B2

000300  BF 15 86 68 81 61 02 CD : 73
000308  38 03 C7 E8 2F E4 AB 43 : EB
000310  60 A0 05 07 A8 28 24 54 : 54
000318  A7 08 09 10 12 EA AB 63 : D2
000320  08 0C 20 61 75 56 3A 64 : FE
000328  DC BC CC F9 AE E3 E3 50 : 21
000330  F9 F0 06 BC EC D9 F3 32 : 95
000338  73 B2 A7 CF 99 3A 6A B2 : 8A
000340  F8 83 73 2F 37 85 29 DE : E0
000348  0E 54 E9 4E CB 55 86 38 : 77
000350  14 47 FA 84 1B 08 36 3D : 6F
000358  F7 18 DF 83 21 B6 B4 8A : 86
000360  82 90 14 9E B0 A4 C5 31 : 0E
000368  56 16 20 A4 41 4B AC AE : 16
000370  8B 83 18 18 DD 75 96 60 : 86
000378  82 27 72 08 97 38 85 8F : 06
-------------------------------------
CKSUM:  44 B0 E7 32 B5 D7.1B 0A BDA4

000380  44 85 5C 4D 7E 1F E3 AB : 9D
000388  1B 20 93 04 9F 9C 24 E4 : 15
000390  53 06 88 00 10 00 E7 2A : 02
000398  93 60 B5 05 AF 4D 5D 1C : 22
0003A0  20 23 FE C0 46 A0 7A B0 : 11
0003A8  27 AC E2 F1 2A B3 28 09 : B4
0003B0  70 4B ED 09 7C 4A 70 CA : B1
0003B8  83 FB 8F EF F4 2A D2 A0 : 8C
0003C0  B9 05 CF DA AF 8B 10 1D : CE
0003C8  20 3B 44 3E 5B 14 55 F1 : 92
0003D0  B2 2A F2 E0 9B 04 DF A2 : CE
0003D8  26 E4 54 06 08 01 90 03 : 00
0003E0  FA D5 89 90 5E 82 F7 AC : 6B
0003E8  B0 8D 10 29 20 53 A7 F7 : 87
0003F0  84 53 55 C7 C8 AC CD 82 : B6
0003F8  7C 13 F2 09 F9 15 21 C2 : 7B
-------------------------------------
CKSUM:  DA 36 C1 86 A8 09 8F 92 64BB

000400  01 E4 03 FA 8A D4 D2 03 : 15
000408  04 18 3B AB 18 D2 90 40 : BC
000410  D5 2F 3B 89 85 F7 F0 AF : E3
000418  A5 7E B0 6E C1 88 BE 3B : 83
000420  EE E2 1F D4 B4 C4 42 A0 : 1D
000428  88 04 47 1C 44 42 A3 7A : 92
000430  7A 9A 25 1A 5A 17 E5 4D : F6
000438  87 E4 48 9C E5 30 39 C2 : 5F
000440  60 7E 14 89 F8 3F 3D 4B : 3A
000448  E7 EC DE 62 9E 2E 0A A0 : 89
000450  55 73 55 79 81 FC 3B 6F : BD
000458  E1 50 21 DB A8 9F 2D 50 : F1
000460  19 04 60 23 39 02 33 12 : 20
000468  91 34 3E B0 FA DF 32 A2 : 60
000470  36 0A E0 57 74 15 99 C1 : 5A
000478  FF 3F C7 FC A2 1C 70 F9 : 28
-------------------------------------
CKSUM:  52 BB A9 A7 27 8C 30 6E D6FD

000480  E7 FD 95 48 6D 01 20 09 : 58
000488  0F 30 48 7A 6A 54 DA 83 : 1C
000490  F1 0F C5 B8 A9 8A 82 C8 : FA
000498  16 5F 4A B7 2C 40 23 FE : 03
0004A0  80 46 80 83 C2 26 DE C7 : 56
0004A8  FA AA A0 B8 25 01 29 E8 : 33
0004B0  09 4C 4A 64 C0 FD C3 F7 : 7A
0004B8  73 D5 51 90 5B 02 DB EC : 4D
0004C0  57 66 A0 80 A3 CB D3 C5 : E3
0004C8  D4 95 B1 3E 0E 33 47 A1 : 81
0004D0  8F E2 9E E3 57 F1 56 34 : C4
0004D8  02 60 E2 9A DD B7 AF 8A : AB
0004E0  A7 93 B7 89 AD 08 2F 13 : 71
0004E8  0E FF 9D 6B 0E 1D F3 D9 : 0C
0004F0  52 5F 56 F1 34 9F 2A 2C : 21
0004F8  52 5F C4 A5 E5 75 6C 16 : F6
-------------------------------------
CKSUM:  08 39 E6 25 67 24 1B 36 7190

000500  A0 B5 F6 C5 AC 2A E7 50 : 1D
000508  20 5E 57 47 21 81 A5 A1 : 04
000510  7F 51 9B 9F 18 35 59 71 : 21
000518  F0 DF 4F 7D 40 32 1E FD : 28
000520  DA E3 32 A0 B9 D7 19 EF : 27
000528  6B 88 46 97 1B B6 55 45 : 3B
000530  76 AE B4 87 17 57 59 5A : 80
000538  8E 75 CC 4E F2 CB 38 80 : 92
000540  D7 06 BF F2 B7 0E 29 66 : E2
000548  0E FA 90 84 87 11 47 B5 : B0
000550  B6 78 8E 1C 3D DE 71 17 : 7B
000558  0F 8F 50 6C 6F 2B D0 68 : 2C
000560  E5 61 BF 55 84 54 14 60 : A6
000568  A3 E6 2A 82 D6 A0 63 07 : 15
000570  6E AA B4 D5 60 1D AC EE : B8
000578  BA A1 8A 1D E9 A5 A0 6B : 9B
-------------------------------------
CKSUM:  D2 6A 83 FB 8F 9F 76 C7 BEE2

000580  BB 9D 78 2A 72 35 76 AF : C6
000588  41 75 FB 49 83 57 25 CD : C6
000590  5B 7A 04 09 DC 97 BA E0 : EF
000598  C5 06 2F E8 B2 8C 10 41 : 71
0005A0  EE C1 06 E7 10 7B D1 63 : 5B
0005A8  0E 25 FF 58 5F 8C AD AC : CE
0005B0  82 78 13 D8 84 F4 8A 8C : 73
0005B8  D1 01 52 02 BD 45 68 6C : FC
0005C0  18 00 C0 DD 58 A7 30 40 : 24
0005C8  CB E2 71 63 0D EC 61 AE : 89
0005D0  8E 94 BD E0 2F 4D 95 57 : 27
0005D8  51 10 11 C6 1B CB 11 D4 : 03
0005E0  C4 AD 81 1F 75 49 14 45 : 28
0005E8  30 BF 7F 1D 99 EE 38 7F : C9
0005F0  C2 E7 5F 26 A9 89 F9 37 : 90
0005F8  55 1D 50 58 02 C3 A2 AD : 2E
-------------------------------------
CKSUM:  38 E7 BE 1D 9B 1D F3 65 5EA3

000600  3A 1E AF 69 93 72 F3 33 : 9B
000608  E6 BB 8F 8D DE 00 00 00 : 9B
000610  00 00 00 00 00 00 00 00 : 00
000618  00 00 00 00 00 00 00 00 : 00
000620  00 00 00 00 00 00 00 00 : 00
000628  00 00 00 00 00 00 00 00 : 00
000630  00 00 00 00 00 00 00 00 : 00
000638  00 00 00 00 00 00 00 00 : 00
000640  00 00 00 00 00 00 00 00 : 00
000648  00 00 00 00 00 00 00 00 : 00
000650  00 00 00 00 00 00 00 00 : 00
000658  00 00 00 00 00 00 00 00 : 00
000660  00 00 00 00 00 00 00 00 : 00
000668  00 00 00 00 00 00 00 00 : 00
000670  00 00 00 00 00 00 00 00 : 00
000678  00 00 00 00 00 00 00 00 : 00
-------------------------------------
CKSUM:  20 D9 3E F6 71 72 F3 33 72A6
```

▶「大魔界村」はそんなに難しくないと思う。１週間くらいでエンディングまでいけた。
吉田　博之(24)福島県

Oh!X LIVE in '94

X68000・Z-MUSIC用 (SC-55対応)	「宇宙刑事ギャバン」より **異世界に光る3本の剣**	Sasaki Tsugutomo 佐々木 嗣朋
X68000・Z-MUSIC用 (SC-55対応)	©KONAMI「究極戦隊ダダンダーン」より **闘え！ ダダンダーン**	Sanuga Eiichi 佐怒賀 英一
X68000・Z-MUSIC ver.2.0用 (SC-55対応)	©KONAMI「メタモルフィックフォース」より **バキャロスの遺跡**	Deai Hirokazu 出合 弘和
X68000・Z-MUSIC ver.2.0用	「スティング」より **エンターテイナー**	Kato Takashi 加藤 隆
X68000・Z-MUSIC ver.2.0用	**中央競馬のファンファーレ2曲**	Okuoka Yoshiyuki 奥岡 良行
X68000・Z-MUSIC用 (SC-55対応)	**ベンザエースのCMソング**	Ohata Keiji 大畑 佳史

今月号は豪華版のLIVE in。特集の記事を読んで何か刺激されるものがあった人は，データを入力して聴くことから始めませんか。比較的短いリストばかりなので面倒くさがりのキミでも大丈夫。慣れたら次は投稿して，Oh!Xの音楽の星を目指すのだ！

大捜索

1曲目は，特撮ドラマ「宇宙刑事ギャバン」より，「異世界に光る3本の剣」をお贈りしましょう。

演奏にはSC-55同等品が必要です。

この作品は，別の作品のオマケとして投稿されたもので，ドキュメントの類はありませんでした。添えられた手紙には「ウルトラマンのサントラ」とあり，唯一の手がかりはデータに書かれていた曲名だけ。さあ，「Oh!X探偵団」(？)の出動です。

復活！ 栄光の東映ヒーロー

ところが……。余裕で見つかるハズの「ウルトラマン」サントラ版CDはどこにいっても見つからず，編集部内でも論議百出。どのウルトラマンかわかりません。作曲者は，作風からして渡辺宙明氏であることが容易に想像できるのですが，初代の「ウルトラマン」ではありません，サントラならば，映画化された「パワード」ではないか，という意見も出ましたが，著作権協会に問い合わせても結局わからずじまい……。

そして，宇宙刑事もので同名の曲があるという情報で，そちらを当たってみたところ，めでたく発見！ 実は「宇宙刑事ギャバン」の曲だったというわけです。

なんでも，作者の佐々木君は友達から渡されたテープを聴いてデータを作ったとのことで，再度確認していただき，我々の推理の正しさが証明されました。う～ん，やはりオマケとはいえ，ドキュメントは書いてくださいね。

ここまでして掲載しようとしたくらいですので，作品はいい味出してます。長さも手ごろですので，特撮ファンならずとも入力してくださいね。

♪守れ，あしたの空を

お次は，ゲームミュージックをお届けしましょう。コナミが贈る戦隊ヒーローもの……もうこれっきゃないですね。「究極戦隊ダダンダーン」より「闘え！ダダンダーン」です。この曲は1ステージの曲ですね。演奏にはSC-55同等品が必要です。

比較的新しいゲーム(半年は経っていますが)ですので，近くのゲーセンにも置いてあるかもしれませんよ。確認がてらに遊びに行くのもいいかもしれませんね。

聴いてみればわかると思いますが，この「闘え！ ダダンダーン」はコナミの音楽としてはかなり異色な存在といえるでしょう。なんといっても，歌詞つきの曲で，ちゃんと「歌われて」いて，しかもその歌手はあの子門真人さんなのです！ 子門さんといえば，「科学忍者隊ガッチャマン」のタイトル曲で有名ですよね。ま，「およげたいやきくん」のほうが有名ですが，この「究極戦

隊ダダンダーン」のイメージするところはレトロなアニメヒーローということですので,「ガッチャマン」のイメージがより近いのです。

作品のデキとしてはかなりイイ線までいってると思いますが,さすがに子門真人の独特な歌声までは再現できていませんね。普通に聴く分には申し分ないのですが……。さすがに難しいレベルでしょう。思わず歌いたくなってしまうノリまで再現されているし,リストも短いので,入力してみんなで歌うっきゃないね!

メタモルフィックフォース

続くゲームミュージックもコナミの作品で,「メタモルフィックフォース」より「バキャロスの遺跡」です。これも同じく1ステージのBGMですね。

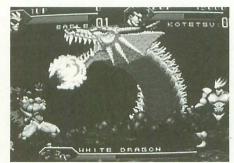
演奏にはZ-MUSIC ver.2.0とSC-55同等品が必要です。まだ買ってない人は手に入れなきゃ聴けないぞ。

恥ずかしながら,私はこのゲームについてはあまり知らないのですが,CDを聴く限りでは,かなりクオリティの高い音楽のようですね。

それにしても,コナミはこのLIVE inへの投稿でも人気がありますねえ。これはやはり「悪魔城ドラキュラ」効果とでも呼べるものなんでしょうか? アーケードゲームにおける音楽に何か惹かれるものがあるからなのでしょうか? ともかく,いまアーケードメーカーのなかで投稿がダントツに多いのは間違いなくコナミでしょう。今回も,さきほどの「究極戦隊ダダンダーン」と「メタモルフィックフォース」の2曲が採用となりました。

この作品では,ギターの旋律をほかの楽器に任せてしまっています。それでいて違和感がないのですから,たいしたものです。こういった手法も割り切っていてなかなかよろしいでしょう。もともとの曲がハードロックを意識して作られているので,ギターが唸ること唸ること(笑)。ちょっと手のうちようがないかも。別の手段としては,うまいアレンジを考えてみるのもひとつの手かもしれませんね。きれいな旋律をもった曲ですので,思い切ってオーケストラっぽくしてみるとか,遊びがいのある曲ではないでしょうか。

究極戦隊ダダンダーン

メタモルフィックフォース

FM音源ピアニスト登場

「エンターテイナー」といえば映画「スティング」ですよね。

演奏はZ-MUSIC ver.2.0が必要ですが,内蔵FM音源だけでOKです。

あいかわらずFM音源のピアノで勝負の加藤君です。なんでも,もともとピアノ曲を究めるためにX68000を買ったんだとか。なるほど,「渚のアデリーヌ」(1993年11月号),「夢路より」(1994年1月号)という流れはそういうところにあったのですか。クラシックのピアノ曲にもどんどん挑戦していただきたいですね。過去にはショパンの練習曲をいくつか投稿していただいてますよね。ちゃんとチェックしてあるんですよ。今後は「ノクターン」などにも手を出してみてはいかがですか? もしくは,人間にはちょっと弾きづらい(?)リストの曲などもいいでしょう。指が届かない曲だって,MMLなら問題ないわけですから。

この作品は曲中でテンポを変えたり,ボリュームにメリハリを出すなど,いろいろと考えられた作り方をしていますね。惜しむらくは,どうしても機械弾きという感じが残っていることでしょう。もし,それが狙いだったとしたら,あまりお勧めはできません。やはり思い切り人間くさいほうがよいと思うのですが,いかがでしょうか。

万馬券祈願!?

ツインクルレースなどでおしゃれ感も出てきて,ほかの公営ギャンブルに比べて人気も急上昇中の競馬ですが,ついにファンファーレの投稿がきてしまいました。ここまでくると,競馬が国民の娯楽として,立派に根づき始めたことを感じざるを得ません。ちょっと大げさかな(笑)。

曲は,中央競馬のファンファーレです。これは地方競馬では違うものなんでしょうかねぇ?

内蔵FM音源だけで再生できます。競馬場に行ったことがある人も,ない人も,その気分を存分に(?)味わってくださいね。

この作品はテクニックうんぬんというより,ネタが勝負のイロモノですので,おおらかな気持ちで受けとめる必要があるでしょう。身近な題材で音楽するという姿勢がさりげなくて面白いですよね。

私としては,競馬関係ならば,ツインクルレースのCMソングの「♪ツインクル・ツインクル……」というWinkの歌のほうが好きなんですけどねぇ。

作者の奥岡さんは,桜花賞の勝ち馬投票券を買いに行った帰りに思いついたそうですが,肝心のレースのほうはどうだったんですか? うまくいっていれば,原稿料と合わせて2度おいしい……なんてことにもなりますかね。

ベンザエース

「♪風邪だ,風邪だ,風邪だ〜」
「♪早め,早め,早め〜」
といえば,どんなCMかすぐ思い当たりま

THE STING

すよね。鷲尾いさ子が出ている風邪薬のテレビコマーシャルです。

今回はこのCMソングを30秒スポットバージョンでお届けしましょう。なんとSC-55用ですが，簡単に内蔵FM音源用に手直しできるでしょう。これもやはりテクニック重視というより，ネタが命のウケ狙いですので，誰にでも入力しやすいように，作品は短いほうがよいでしょう。もちろん，テクニックバリバリの短い曲というのも面白い素材ではあるのですが。

この程度の長さの曲から耳コピを始めるというのは手ごろでいいですよね。そういった意味では，CMソングというのは練習曲の宝庫なのかもしれません。つい口ずさんでしまう曲なんかがベストでしょう。メロディがしっかりした曲ほど作りやすいものです。

ところで，このCMは関東ローカルなんでしょうか？　できれば全国区のCMのほうがウケがいいですよね。　(SIVA)

リスト1　異世界に光る3本の剣


```
  1: /
  2: /
  3: /
  4: .comment 宇宙刑事ギャバン 「異世界に光る三本の剣」 (SC-5
5) by tusgu.
  5:
  6: (i)
  7: (b1)
  8:
  9: (m1,2000)(amidi1,1)
 10: (m2,3000)(amidi2,2)
 11: (m3,2000)(amidi3,3)
 12: (m4,2000)(amidi4,4)
 13: (m5,2000)(amidi5,5)
 14: (m6,2000)(amidi6,6)
 15: (m7,2000)(amidi7,7)
 16: (m8,2000)(amidi8,8)
 17: (m10,2000)(amidi10,10)
 18: (m11,2000)(amidi10,11)
 19: (m12,2000)(amidi10,12)
 20:
 21: .roland_exclusive $10,$42={$40,$00,$7f,$00}
 22:
 23: (t1)  @i$41,$10,$42 x$40,$01,$30,3 x$40,$01,$38,4
 24: (t2)  @i$41,$10,$42
 25: (t3)  @i$41,$10,$42
 26: (t4)  @i$41,$10,$42
 27: (t5)  @i$41,$10,$42
 28: (t6)  @i$41,$10,$42
 29: (t7)  @i$41,$10,$42
 30: (t8)  @i$41,$10,$42
 31: (t10) @i$41,$10,$42
 32: (t11) @i$41,$10,$42
 33: (t12) @i$41,$10,$42
 34:
 35: / Strings.-----------------------------------------
 36: (t1) t130@49@u127@v110@q1@p90o6l16 @e74,40 r2
 37:
 38: (t1) |:|:c_15c~d_d~e-_e-~f_f~e-_e-~f_f~e-_e-~d_d~:|
 39: (t1) |:|:c_c~d_d~e-_e-~f_f~e-_e-~f_f~e-_e-~d_d~:|
 40: (t1) c_c~d_d~e-_e-~f_f~g-_g-~f_f~e-_e-~d_d~
 41: (t1) |e-_e-~f_f~e-_e-~f_f~e-_e-~f_f~e-_e-~d_d~:|
 42: (t1) e-_e-~f_f~e-_e-~f_f~e-_e-~f_f~f_f~g-_g-~
 43: (t1) ~|:16q6frq4ff:|
 44: (t1) @q1e-2.q6e->@q1b-rb-<c2c2d-2.q6d->@q1a-ra-b-2b-2<
 45: (t1) |:c2.q6c>@q1grg<:||c1&c1_:|
 46: (t1) c1 >>r4crccc8
 47:
 48: / Strings.-----------------------------------------
 49: (t2) @49@u118@v110@q1@p70o5l16 @e74,40 r2
 50:
 51: (t2) |:|:c_15c~d_d~e-_e-~f_f~e-_e-~f_f~e-_e-~d_d~:|
 52: (t2) |:|:c_c~d_d~e-_e-~f_f~e-_e-~f_f~e-_e-~d_d~:|
 53: (t2) c_c~d_d~e-_e-~f_f~g-_g-~f_f~e-_e-~d_d~
 54: (t2) |e-_e-~f_f~e-_e-~f_f~e-_e-~f_f~e-_e-~d_d~:|
 55: (t2) e-_e-~f_f~e-_e-~f_f~e-_e-~f_f~f_f~g-_g-~
 56: (t2) ~|:16q6frq4ff:|
 57: (t2) @q1e-2.q6e->@q1b-rb-<c2c2d-2.q6d->@q1a-ra-b-2b-2<
 58: (t2) |:c2.q6c>@q1grg<:||c1&c1_:|
 59: (t2) c1 >>r4crccc8
 60:
 61: / Brass.-------------------------------------------
 62: (t3) @62@u124@v127@q2@p50o5l16 @k-2 @e74,40 r2
 63:
 64: (t3) |:r1r1
 65: (t3) @v86|:8c&~5:|c4 @v120c*14>gr*10<d*14 c1 r1 r1
 66: (t3) |:c*46r*2c*14>gr<d*10:| c1 r1 r1
 67: (t3) f2.f*14crf*8r*2 f1 f2.f*14crf*10 a-*35r*13f2.
 68: (t3) r*1344|r1:| >>r4crccc
 69:
 70: / Brass.-------------------------------------------
 71: (t4) @57@u127@v115@q4@p40o5l16 @k1 @e65,40 r2
 72:
 73: (t4) |:r1r1@u127
 74: (t4) @v86|:8c&~5:|c4 @v120c*14>gr*10<d*14 c1 r1 r1
 75: (t4) |:c*46r*2c*14>gr<d*10:| c1 r1 r1
 76: (t4) f2.f*14crf*9r*1 f1 f2.f*14crf*10 a-*35r*13f2.
 77: (t4) @u108q6e-2.e->b-rb-<c2c2d-2.d->a-ra-b-2b-2<
 78: (t4) |:c2.c>grg<:||c1&c1:| c1 >>r4crccc
 79:
 80: / Bass.--------------------------------------------
 81: (t5) @46@u127@v110q7p3o2l16 @e60,40 r2
 82:
 83: (t5) |:|:8crcc:||:32crcc:|
 84: (t5) |:16>frff<:|r*1152||:8crcc:|:||:4crcc:| r4crccc4
 85:
 86: / Piano.-------------------------------------------
 87: (t6) @2@u127@v110q7p3o4l16 @e55,40 r2
 88:
 89: (t6) |:|:@u118'ce-g2',1@u110'ce-g4'@u122'ce-g4':|
 90: (t6) |:|:@u118'ce-g2'@u110'ce-g4'@u122'ce-g8'r@u110'ce-g

 91: (t6) |@u118'ce-g2'@u110'ce-g4'@u122'ce-g4':|
 92: (t6) @u98'ce-g4'@u108'ce-g4'@u118'ce-g4'@u125'ce-g4':|
 93: (t6) @u117|:|:'f>a-4':||:'e->a-4':||:'d>a-4':||:'d->a-4'
:|:|
 94: (t6) 'e->b-2.''e->b-''>b-g'r'>b-g''c>g2''c>g2'
 95: (t6) 'd->a-2.''d->a-''>a-f'r'>a-f''>b-f2''>b-f2'
 96: (t6) |:'c>g2.''c>g''>ge-'r'>ge-':|
 97: (t6) @u80||:8u+5'ce-g8''ce-g''ce-g':|:|
 98: (t6) @u119 'ce-g1' r4'c>ge-8''c>ge-''c>ge-''c>ge-4'
 99:
100: / Brass.-------------------------------------------
101: (t7) @59@u120@v119q7@p76o2l16 @e75,40 r2
102:
103: (t7) |:r|r1 r*1536 |:f2e-2d2d-2:| r*1344|r1:|
104: (t7) r4crccc
105:
106: / Brass.-------------------------------------------
107: (t8) @58@u120@v115q7@p93o4l16 @e75,40 r2
108:
109: (t8) |:r1q7r1 r*1536 |:f2e-2d2d-2:|
110: (t8) q6e-2.e->b-rb-<c2c2d-2.q6d->a-ra-b-2b-2<
111: (t8) |:c2.c>grg<:||c1&c1:| c1 >r4crccc
112:
113: / Drums.-------------------------------------------
114: (t10) @49@u80@v127q8p3o2l16 @r1 @e90,40
115: (t10) @y28,38,64 r2
116:
117: (t10) |:|:|:3@u89dr@u95d@u100d:|@u95dd@u100dd:|
118: (t10) |:12|:3@u89dr@u95d@u100d:|@u95dd@u100dd:|
119: (t10) |:12@u79dr@u85d@u90d@u85dd@u90dd:|
120: (t10) ||:|:3@u89dr@u95d@u100d:|@u95dd@u100dd:|:|
121: (t10) |:3@u89dr@u95d@u100d:|@u95dd@u100dd r4crccc
122:
123: / Drums.-------------------------------------------
124: (t11) @u127o2l4 @r1 @y28,41,64 @y24,41,59 r2
125:
126: (t11) |:frfrfrfr
127: (t11) |:frfrfrfr frfrffff:|
128: (t11) frfrfrfr frfrfrf{rrff} |:3f1frf{rrff}:|||:frfr:|:|
frfrr{frff}f
129:
130: / Drums.-------------------------------------------
131: (t12) @u116o3l4 @r1 @y28,59,64 r2
132:
133: (t12) |:r1b1 |:r1r1b1r1:| r1r1r1r1
134: (t12) b1b2b2 b1b2b2 b1b1 b1r1:|
135:
136: (p)
```

リスト2　異世界に光る3本の剣のカウンタ表示

```
 1:00002118 00000000   2:00002118 00000000   3:00002110 00000000   4:00002110 00000000
 5:00002130 00000000   6:00002130 00000000   7:0000210C 00000000   8:0000210C 00000000
10:0000210C 00000000  11:00002130 00000000  12:00002160 00000000
```

リスト3　闘え！ ダダンダーン

```
1: /----------------------------------------------------------
2: /             composed by コナミ矩形波倶楽部
3: /
4: /                    1994/02/18
5: /
6: /                       programmed by E.S
7: /----------------------------------------------------------
8: .comment 究極戦隊ダダンダーンTM by コナミ矩形波倶楽部
(GS音源)
9:
```

▶なんで新しい68がでんのじゃあ!　　神田　幸男(18)神奈川県

```
 10: /---- INIT ----
 11: (i)
 12: (b1)                        /MIDI'1-16'
 13: (o180)
 14:
 15: /---- TRACK SETUP ----
 16: (M1,2000)(AMIDI1,1)
 17: (M2,2000)(AMIDI2,2)
 18: (M3,2000)(AMIDI3,3)
 19: (M4,2000)(AMIDI4,4)
 20: (M5,2000)(AMIDI5,5)
 21: (M6,2000)(AMIDI6,6)
 22: (M7,2000)(AMIDI7,7)
 23: (M8,2000)(AMIDI8,8)
 24: (M9,2000)(AMIDI10,9)
 25: (M10,2000)(AMIDI10,10)
 26: (M11,2000)(AMIDI10,11)
 27: (M12,2000)(AMIDI10,12)
 28:
 29: /---- MIDI SETUP GS ----
 30: .roland_exclusive $10,$42={$40,$00,$7f,$00}
 31: .sc55_v_reserve $10={1,4,3,1,2,5,1,1,1,4,0,0,0,0,0,0}
 32: .sc55_reverb $10={3,6,6,40,40,80,00}
 33: .sc55_chorus $10={3,6,40,20,10,50,80,00}
 34:
 35: /---- MML SETUP ----
 36:
 37: /---- TRACK 1 VOCAL ----
 38: (t1) @30 o4 l8 q8 @u127 @v80 r4 @p64 [K.SIGN +c,+f,+g]
 39: (t1) |:8r1:|
 40: (t1) r1[do]ag4f&ffffge4e&e4r4ffffed4c&
 41: (t1) c2.r4>b4<df&f4@56@v100<f>@v80@30@k0,192,8f4@k
 42: (t1) agf&f4@56@v100<f4>@v80@30q6g_20g~20gfq8f!f4g&
 43: (t1) g2.r4ag4f&fff_20f~20g_20e&e4~20r2q6ffffq8ed4c&
 44: (t1) c2.r4>b4<df&f4@56@v100<f>@v80@30@k0,64,8g4@k
 45: (t1) agf&f4@56@v100<f>@v80@30@k0,128,8f4@kfgff!g4f&
 46: (t1) f2.r4r4fff4rffg4g&ggfga4b@k0,128,8b2&b8&b1@k
 47: (t1) r4bbbbbaagr4{gab}2<c1
 48: (t1) r2.>b<cd2.c@k0,-128,8c4&c8@k0
 49: (t1) r4>{b_20a~20g}2a4r4aab<c&
 50: (t1) c2.r4d4ddrdc>b<c4r4cc_20c~20c>f2.r4
 51: (t1) r1r1r1[loop]
 52:
 53: /---- TRACK 2 BRASS ----
 54: (t2) i8 @62 o4 l8 @u127 @v60 r4 @p80 [K.SIGN +c,+f,+g]
 55: (t2)  @I$41,$10,$42 @E2,0
 56: (t2) q6'f>a''g4>b''ac'&'ac''f>a''g>b''ac'
 57: (t2) 'g>b''a4c''bd'&'bd''g>b''ac''bd'
 58: (t2) r4'a4d'q8{'bf''af''bf'}2'<c1>f'
 59: (t2) r4<'c4>gf!c'{'d+>bge''e>gfc''f!>bge'}2 @k0,-64,40
 60: (t2) 'f4c>af'r2.@kr1r1
 61: (t2) r1[do]r1r1r1 @u110r4cr>ba&a4r1r1r4c!2.
 62: (t2) c1r4.<'fc>af'r2r4.'e>bge'&'e2>bge''d>afd'r8'd2.>afd

 63: (t2) r4'c>a'r>'bg''ae'&'a4e'r1r1r1
 64: (t2) r1r4.<'f16d>b''f16d>b''f4d>b'r4r4.
 65: (t2) >'bge'&'b2ge'r4<'e2.c>a'
 66: (t2)   r2'c>a+f''c4.>af'r4.'f16d>b''f16d>b''f4d>b'
 67: (t2) r4r2>{'bge''bge''bge'}2r4(a4<f){fge}2
 68: (t2) c2>a+16b16<cc16d16er4.'f16d>b''f16d>b''f4d>b'
 69: (t2) r4r4.'fd>b'&'f2d>b''c2>af'frga&
 70: (t2) a2.r>d16f16b4r4b4r4<c>{cfg}8'b2.ge'f1
 71: (t2) r2r'g4ec''bge'<'c2>af'&'c>af'>'b4.ge'<
 72: (t2) l4'c>af''e>bge''e>bge''e>bge'[loop]
 73:
 74: /---- TRACK 3 BRASS ----
 75: (t3) i8 @62 o3 l8 @u127 @v60 r4 q8 @p80 [K.SIGN +c,+f,+g
]
 76: (t3) f1 e1 r4 d4<{'fd>b<''fd>b<''fd>b<'}2 g1
 77: (t3) r4c4{ccc}2 f4r2. @u110r1r1r1[do]|:3r1:|r1r1r1r4d+2.
 78: (t3) f!1r1r1r1 |:4r1:|
 79: (t3) r1r4.'f16d>b''f16d>b''f4d>b'r4r4.>'bge'&'b2ge'r4<'e
2.c>a'
 80: (t3) 'c16>a+f''c16>af''c>af'|:2'c16>af':|'c>af'r2
 81: (t3) r4.|:2'f16d>b':|'f4d>b'r4r2
 82: (t3) >{'bge''bge''bge'}2r4(a4<f){fge}2
 83: (t3) c2>a+16b+16<cc16d16er4.'f16d>b''f16d>b''f4d>b'
 84: (t3) r4r4.'fd>b'&'f2d>b''e2c>a'frga&
 85: (t3) a2.r>c16f16<l4'fd>b'r'fd>b'r'g8f!c'{cfg}8'g2.f!c'r1
 86: (t3) r1r1l4r'gf!c''gfc''gfc'[loop]
 87:
 88: /---- TRACK 4 FAZY GUITAR ----
 89: (t4) @31 o3 l8 @u120 @v65 r4 @p64 [K.SIGN +c,+f,+g]
 90: (t4) @I$41,$10,$42 @E4,0 @M40
 91: (t4) |:5r1:|r4f4fef@k0,64,16b@k
 92: (t4) rbrar<cef&f@k0,160,16e4e4@kece
 93: (t4) >b16a16faa@k0,160,16b@k<cfc [do]|:32f32&_1:|r1r1 @v
75
 94: (t4) >r@k0,128,8b4@k<c>@k0,128,16 b2@kr1r1r1|:4r1:|
 95: (t4) r1>b1f4.<@k0,128,4b@krba4g2c4.@k0,128,8b4@k
 96: (t4) <cfc|:2@k0,128,4e4@k:|
 97: (t4) >>b1e2<{bab}2|:2@k0,128,4b4@k:|<c>bab
 98: (t4) <c@k0,128,8e@kr@k0,128,8e@k
 99: (t4) fe4d16c16>b4b4<c>babb<crdrd+re
100: (t4) >|:2@k0,128,4b4@k:|<@k0,128,4c4@k>a<c&
101: (t4) c2>f16g16a+a16b16<cr2<fefa|:2@k0,128,8b4@k<c:|
102: (t4) >@k0,128,4b@k<cr4f4fred+&
103: (t4) d+1r1r4>@k0,128,8b4@k<c2r4>>>     f4fef@k0,160,16b@k
104: (t4)     rbrar<ce@k0,128,8e4@k@k0,160,16e@kf@k0,160,16e@k
105: (t4) fece>b16a16faa@k0,160,16b@k<cfc[loop]
106:
107: /---- TRACK 5 STRINGS ----
108: (t5) @49 o3 l16 @u127 @v60 r4 @p40 [K.SIGN +c,+f,+g]
109: (t5) r4fga<cf2>r4ga<cfg2 @v60
110: (t5) r4<'a4>a'{'b>b''a>a''b>b'}2 '<c1>c'
111: (t5) r4l4<'c>c'{'d+>d+''e>e''f!>f!'}2 >f>>r
112: (t5) @u110l2'bg''<c>a''bg''af''<c>g'
113: (t5) <'f>a''gc' [do]'fc''af''ge''bg'<'d1>a'
114: (t5) 'c4>a'l8'<c>a'r'bg''ae'&'a4e'>'b1f''<c1>f'>
115: (t5) l4g<'c!>g''e-c>g''gec>g'
116: (t5) '<c>c''bd+''ae''gf!'l2<f'af''ge''bg'<'d1>a'
117: (t5) l8'c>a'r'c>a'r>'bg''af'&'a4f''b1f''<c1>f'>l4g<d+cg
118: (t5) f>cdef1g1aa8e8&e2 <e2dc>b2.ag2<{fge}2c1&
119: (t5) c2.rd2.c>b2{bab}2<c2l8frga& a1<l4brbr<c8r8c2.>f1
120: (t5) >f1>'a2f'<'c2g''f>a'r2.[loop]
121:
122: /---- TRACK 6 CHOIR&GUT. ----
123: (t6) @53 o3 l4 @u127 @v50 r4 @p40 [K.SIGN +c,+f,+g]
124: (t6) @I$41,$10,$42 @E6,0
125: (t6) '<c1>af''b1ge'r4'a4fd'{'fd>b<''fd>b<''fd>b<'}2c1
126: (t6) r4<'gfc'{'gfc''gfc''gfc'}2'c>af' @25 q1
127: (t6) l16a8aaa8aaa8aa |:2'a8>a'aaa8aa'a8>a'aaa8aa:|
128: (t6) 'a8>a'aaa8aa'a8>a'aaa8aa q8 l1
129: (t6) [do]@53 'ac>af''g>bge''f>afd'
130: (t6) 'e>ec>a''d>fd>b''c>c>ae'
131: (t6) 'c!>e-c!>g''c>gf!c''ac>af''g>bge''f>afd'
132: (t6) 'e>ec>a''d>fd>b''c>c>af'
133: (t6) 'c!2e-c!>a+g''c2>ag!c'
134: (t6) 'c>af'>'fd>b''bge''ec>a'<'c>af'>'fd>b''bge''ec>a'
135: (t6) '<c>af''fd>b''fd>b'<'c2>af''c4>af''>b8ge''c8>af'&
136: (t6) 'c>af'l4'fd>b'r'fd>b'r'g8f!c'r8'f2.fc''c>af' @25 q1
137: (t6) l16a8aaa8aaa8aa
138: (t6) |:4'a8>a'aaa8aa:|'f8>f'ffr2.[loop]
139:
140: /---- TRACK 7 BASS ----
141: (t7) @35 o1 l8 @u127 @v60 r4 @p64 [K.SIGN +c,+f,+g]
142: (t7) @I$41,$10,$42 @E7,0
143: (t7) f4r2.e4r2.r4d4{bgb}2<|:8c:|
144: (t7) >crc4<{ccc}2>f4.f>bb4b<|:2f4.fccec:|
145: (t7) f4.fccec [do]a4.cff4fe4.eee4ed4.add4d
146: (t7) aarc&cdd+eb4.ff4bcf4.cc4fcg4.d+d4gd
147: (t7)   <cc>bbaaggf4.ffcff!e4.eee4ed4.add4d
148: (t7) aarc&cdd+eb4.ff4b>b<f4.cc4fcg4.gc4c4
149: (t7) fffb16<c16ffe16f16f>>bb<rfrbrfeerbr<er>baar<erar>a
150: (t7) ffra+rb<c!c>>bb<rfrbreee<@k0,-256e4@k>{eee}2aacceeg
!g
151: (t7) a+aa16aa16f16ff16a<c>>bbr<frbrfggr<drgrdf2fred+&
152: (t7) d+2>d+4d4b>bbb<b>bbb<<cr>c2.f4r4re4.
153: (t7) f4r4rcecfcfcfeeefr<|:3@k0,-256c4@k:|[loop]
154:
155: /---- TRACK 8 TIMPANI ----
156: (t8) @48 o2 l8 @u115 @v80 r4 @p64 [K.SIGN +c,+f,+g]
157: (t8) @I$41,$10,$42 @E8,0
158: (t8) f4_40f16f16f16f16f4r4~40e4_40e16e16e16e16e4r4r4d4{d
dd}2
159: (t8) <~40c>_40g<~40c>_40g<~40c>_40g<~40c>_40g
160: (t8) ~40a4a4{aaa}2|:4r1:| [do]|:16r1:|r4.b16b16b2r1r1
161: (t8) |:2e16e16e:|eee4r4.b16b16b2r2{eee}2b2r2
162: (t8) |:2e16e16e:|ee4.r4.b16b16b2r1f4r4f4ff
163: (t8) r1b4>b4<b4>b4<c4c4r4r4l16ffff8|:6c:|
164: (t8) r4ffff8|:6c:|l8|:4fc:|l4fcccc[loop]
165:
166: /---- TRACK 9 CONGA ----
167: (t9)  @1 o3 l8 r4 @I$41,$10,$42 @E10,0 @u100 @v80
168: (t9) r1r1 a+2 r2|:5r1:|
169: (t9) <r1[do]|:3|:2d+dre16e16:|:|      |:4|:2d+dre16e16:|
:|
170: (t9)       r1|:3|:2d+dre16e16:|:|      |:4|:2d+dre16e16:|
:|
171: (t9)   |:2r1|:3|:2d+dre16e16:|:|:| r1 |:2|:2d+dre16e16:|
:| r1
172: (t9) |:4r1:||:3r1:|[loop]
173:
174: /---- TRACK 10 CYMBAL ----
175: (t10) @1 o2 l4 r4 @I$41,$10,$42 @E10,0 @u100 @v80
176: (t10) r4|:7f+:|r2<{d-d-d-}2 d-r2.r2{d-d-d-}2d-r>l8|:20f+
:|
177: (t10) |:6f+:|b-4 [do]<d-2>|:20f+:||:32f+:| r1<d-2>|:53f+
:|
178: /(t10) r1<d-2>|:20f+:|r1<d-2>|:6f+:|b-8<{d-d-d-}2d-2>|:4
f+:|
179: (t10) r2r4r8<d-2&d-8>|:20f+:|r2r4r8<d-2&d-8>|:6f+:|f+8<{
d-d-d-}2d-2>|:4f+:|
180: (t10) r1<d-2>|:12f+:| l4f+f+<d-r8d-8
181: (t10) r1d-2d-2d-4d-2.d-2>l8rf+rf+
182: (t10) <d-2>rf+rf+|:4f+b-:|b-f+<d-4>f+4<d-4>[loop]
183:
184: /--------------------------------------------------------
185: /---- TRACK 11 SNARE&TOM ----
186: (t11) @1 o2 l4 r4 @I$41,$10,$42 @E10,0 @u100 @v80
187: (t11) dr2.dr2.d8{<d>bg}8d{rrr}2r2l16dd<dd>bbgg
188: (t11) d8l16{<d>bg}8d4{rrr}2 l4|:3rdrd:|
189: (t11) rdrd [do]|:6rdrd:|rdrd8d16d16
190: (t11) |:3d8d16d16:|d8d8rd8d8rd|:5rdrd:|rdrd8d16d16
191: (t11) d8d16d16d8<d16d16d8>b16b16b8g8 rdrd16d16d8
192: (t11) rd8d8rd  rdrd16d16d8
193: (t11) <d16d16d8>b16b16b8g16g16g8d8d8 rdrd16d16d8
194: (t11) rd{rrr}2 rdrd16d16d8
195: (t11) <d16d16d8>b16b16b8g16g16g8d8d8 rdrd16d16d8
196: (t11) rd8d8rdr8{<d>bg}8ddd
197: (t11) r8l16dddddd<dddd>bbggl4<d>b<d>g dr2. dddd
198: (t11) dddd|:6d8d8:|l16dddd<dd>gd[loop]
199:
200: /---- TRACK 12 BASS ----
201: (t12) @1 o2 l4 r4 @I$41,$10,$42 @E10,0 @u100 @v80
202: (t12) cccc cccc cc{ccc}2 cccc
203: (t12) cc{ccc}2    |:3crc8c8r:|
204: (t12) crc8c8r [do]|:7crc8c8r:|
205: (t12) crc8c8rcrcr |:6crc8c8r:|
206: (t12) crc8c8rc8c8cr8c8rc8c8rc8c8rc8c8rr8c8r
207: (t12) |:2c8c8rr8c8r:|c8c8r{ccc}2 c8c8rr8c8r
208: (t12) |:2c8c8rr8c8r:|c8c8rc8c8r ccr.c8
209: (t12) r8c8rcr l8rcccrccc l4rccccrr8c8r
210: (t12) |:3crr8c8r:|[loop]
211:
212: /---- PLAY ----
213: (p)
```

▶7月号も付録ディスクつきか。そろそろ定期購読をしたほうがいいかもしれない……。
大久保　明弘(21)岩手県

リスト4　闘え！ ダダンダーンのカウンタ表示

```
 1:000006F0 00001980   2:000006F0 00001980   3:000006F0 00001980   4:000006F0 00001980
 5:000006F0 00001980   6:000006F0 00001980   7:000006F0 00001980   8:000006F0 00001980
 9:000006F0 00001980  10:000006F0 00001980  11:000006F0 00001980  12:000006F0 00001980
```

リスト5　バキャロスの遺跡

```
  1: .comment - メタモルフィックフォース「バキャロスの遺跡」
SC-55 (C)KONAMI  By DMD.
  2:
  3: /        1994-2-19
  4: /        Z-MUSIC Ver2.00
  5:
  6: (I)
  7: (b1)
  8: (M1,2000)(aMidi1,1)
  9: (M2,2000)(aMidi2,2)
 10: (M3,2000)(aMidi3,3)
 11: (M4,2000)(aMidi4,4)
 12: (M5,2000)(aMidi5,5)
 13: (M6,2000)(aMidi6,6)
 14: (M7,2000)(aMidi7,7)
 15: (M8,2000)(aMidi8,8)
 16: (M9,2000)(aMidi9,9)
 17: (M10,2000)(aMidi10,10)
 18: (M11,2000)(aMidi10,11)
 19: (M12,2000)(aMidi10,12)
 20: (M13,2000)(aMidi11,13)
 21: (M14,2000)(aMidi12,14)
 22: (M15,2000)(aMidi13,15)
 23:
 24: /        SC55 SYSTEM SETUP
 25:
 26: .SC55_INIT
 27:
 28: .SC55_PRINT "METAMOR PHIC FORCE"
 29:
 30: .SC55_V_RESERVE ={2,2,2,2,2,2,3,2,2,4,2,2,3,1,1,0}
 31:
 32: .SC55_REVERB ={4,7,80,20,100,50,40}
 33:
 34: .SC55_CHORUS ={4,7,100,20,80,6,20,40}
 35:
 36: /        MML DATA
 37:
 38: (t1)     R2[DO]
 39: (t2)     R2[DO]
 40: (t3)     R2[DO]
 41: (t4)     R2[DO]
 42: (t5)     R2[DO]
 43: (t6)     R2[DO]
 44: (t7)     R2[DO]
 45: (t8)     R2[DO]
 46: (t9)     R2[DO]
 47: (t10)    R2[DO]
 48: (t11)    R2[DO]
 49: (t12)    R2[DO]
 50: (t13)    R2[DO]
 51: (t14)    R2[DO]
 52: (t15)    R2[DO]
 53:
 54: /        D.GUTIAR 1--------------------------------------
 55: (t1)     T182i8@31o2@U100@v100q8@p6318@M100@H24@S5,6
 56: (t1)     @G12@e40,30[k.sign+f]
 57: (t1)     |:7eeeeeeee:|e'<da'ee'<cg'e'b<g'e
 58: (t1)     |:3eeeeeeee:|e<c>bagab<c
 59: (t1)     |:3eeeeeeee:|eeee'<c4g4''b4<a4''a1<e1''b1<a1'
 60: (t1)     '<c1g1''<d1a1''<e1b1'<(a*12b)&a2...@v85ee'<e4b4'
 61: (t1)     eeeeee'<e4b4'e'<eg'ee
 62: (t1)     o3cc'<c4g4'cccc'd4.a4.''c4g4'>a<c>b
 63: (t1)     o2|:ee'<e4b4'eeee:|o3|:cc'<c4g4'cccc:|
 64: (t1)     o2ee'<e4b4'eeeeee'<e4b4'e'<eb'ee
 65: (t1)     o3cc'<c4g4'cccc'd4.a4.''c4g4'>a<c>b
 66: (t1)     o2|:ee'<e4b4'eeee:|
 67: (t1)     o3cc'<c4g4'cccc
 68: (t1)     o2ee'<e4b4'>afge<|:'cg''d2..a2..'ddddddd:|
 69: (t1)     'da''e2..b2..'>eeeeeeee<'da''e*120b*120''da''eb'
 70: (t1)     'g4.<d4.''f*120<c+*120''cg''d2..a2..'ddddddd
 71: (t1)     'gc''a*264d*264'l16>defefgab
 72: (t1)     l8<'g4.c4.''a*120>b*120'
 73: (t1)     dc>bbagaa+'b4.<f4.'<'c*120g*120'c>babagaa+
 74: (t1)     <'g4.c4.''f*120>b*120'dc>bbagaa+
 75: (t1)     'b4.<f4.'<'g4.c4.''a4.d4.''g4.c4.'r2
 76: (t1)     >|:4bbbbbbbb:|
 77:
 78: /        O.GUTIAR 1--------------------------------------
 79: (t2)     i0@30o5@U117@v117q8@p63116@M210@H24@S5
 80: (t2)     @G12@e60,0[k.sign+f]
 81: (t2)     |:4@w1:|<(d*12e)&d2...r1
 82: (t2)     r1(a*24b)&a2..|:13@w1:|
 83: (t2)     (a*24b)&a2..|:6@w1:|
 84: (t2)     r4(d*24e)&d2&d8r4(e*24f)&e2&e8
 85: (t2)     |:6@w1:|
 86: (t2)     r4(d*24e)&d2&d8r4(e*24f)&e2&e8
 87: (t2)     (c*24d)&c2..
 88: (t2)     >a<dc>ba<c>bagbagfedf
 89: (t2)     dfegdag<d>b<c>b<dcedf
 90: (t2)     afdbfd<c>fdbfd<c>d<d>d
 91: (t2)     (a*24b)&a8b8(a*24b)&a2
 92: (t2)     {>>b<egbegbge>bgeegb<egb<e>bge>bg}1
 93: (t2)     <(a*24b)&a8<e8>(a*24b)&a4d8e8
 94: (t2)     (f*24g)&f4(e*24f+)&e2r1r1
 95: (t2)     |:3@w1:|>bagab<cdecdefgfab&b8a
 96: (t2)     g&gfe8dc8>baga8&adefgebg&gfgefdc>b
 97: (t2)     <<(b*24<c)c>(b*24<c)c>(a*24b)&a2
 98: (t2)     (c*24d)g(c*24d)a(c*24d)
 99: (t2)     bedc>bab<c(e*24f)&e4(f*24g)&f4
100: (t2)     (g*24a)&g4(b*24<c)&b4
101: (t2)     'f>b''e>a''d>g''c>f'>'be''ad''gc''f>b'|:4@w1:|
102:
103: /        D.GUTIAR 2--------------------------------------
104: (t3)     i8@31o2@U100@v100q8@p31116@M100@H24@S5,5
105: (t3)     @G12@e40,30[k.sign+f]
106: (t3)     b1&b1bbbbr2.l8rc>bagab<cb1&b1
107: (t3)     l16|:bbbbr4:||:23@w1:|
108: (t3)     |:6@w1:|
109: (t3)     i0@30o5@v107@p63r1r4<(c*12d)&c2&c8.|:24@w1:|
110:
111: /        D.GUTIAR 3--------------------------------------
112: (t4)     i8@31o2@U100@v100q8@p95116@M100@H24@S5,5
113: (t4)     @G12@e40,30[k.sign+f]
114: (t4)     e1&e1eeeer2.l8rc>bagab<ce1&e1
115: (t4)     l16|:eeeer4:||:23@w1:|
116: (t4)     |:28@w1:|
117: (t4)     @49o3b1&b1&b1&b1
118:
119: /        STRINGS 1---------------------------------------
120: (t5)     i0@46o3@U80@v80q8@p6318@G12@e40,20[k.sign+f]
121: (t5)     '<b<e'ee'<a<d'ee'<g<c'e
122: (t5)     e'<a<d'ee'<b<e'e'<<cf!'e
123: (t5)     '<b<e'ee'<a<d'ee'<g<c'e
124: (t5)     e'<a<d'ee'<b<e'e'<fb'e
125: (t5)     '<b<e'ee'<a<d'ee'<g<c'e
126: (t5)     e'<a<d'ee'<b<e'e'<<cf!'e
127: (t5)     '<b<e'ee'<a<d'ee'<g<c'e
128: (t5)     <r'a<d'r4'g<c'r'ab'r
129: (t5)     o5|:'e<e'rr'd<d'rr'c<c'rr'd<d'rr'e<e'r'f!<f!'r
130: (t5)     'e<e'rr'd<d'rr'c<c'rr'd<d'rr'c<c'r'>b<b'r:|
131: (t5)     r1r1
132: (t5)     ra<e>a<d>a<c>a<g>a<f!>a<e>a<d>ar1r1
133: (t5)     @49'>e*384>b*384'
134: (t5)     o4e1d4.c8&c2e1&e1&e1f1
135: (t5)     <e1&e2&edefg1f4.e&e2e1&e2&edef
136: (t5)     g2&gefgr4a2.<drrcr4>brr<cr4drcr
137: (t5)     drrcr4>brr<cr4drdr
138: (t5)     @46@v100|:4@w1:|
139: (t5)     drrcr4>brr<cr4drcr
140: (t5)     drrcr4>brr<cr4>d2|:11@w1:|
141: (t5)     @49>d1
142:
143:
144: /        STRINGS 2---------------------------------------
145: (t6)     i0@46o3@U80@v80q8@p6318@G12@e40,20[k.sign+f]
146: (t6)     |:8@w1:|
147: (t6)     |:'<eb'ee'<da'ee'<cb'ee'da'ee'<eb'e'<f!<c!'e
148: (t6)     '<eb'ee'<da'ee'<cg'ee'<da'ee'<cg'e'b<f'e:|
149: (t6)     o4r'a<a''<e<e''a<a''<d<d''a<a''<c<c''a<a'
150: (t6)     '<e<e''a<a''d<d''a<a''<c<c''a<a''<d<d''<e<e'
151: (t6)     ra<e>a<d>a<c>a<g>a<f!>a<e>a<d>a@49r1r1>g1&g1
152: (t6)     o4c1>a4.g&b2b1&b1&b1&b1
153: (t6)     b1&b1<c1d4.c&c2>b1&b1
154: (t6)     b1&r4b2.<arrgr4frrgr4argr
155: (t6)     arrgr4frrgr4arar
156: (t6)     @46@v100|:4@w1:|
157: (t6)     arrgr4frrgr4argr
158: (t6)     arrgr4frrgr4a2|:10@w1:|
159: (t6)     @49a1&a1
160:
161: /        STRINGS 3---------------------------------------
162: (t15)    i0@46o3@U80@v80q8@p3118@G12@e40,20[k.sign+f]
163: (t15)    |:24@w1:|
164: (t15)    g1r1g1&g1&g1&g2<l16defgab<cd
165: (t15)    o3g1&g1g1a4.g8&g2g1&g1
166: (t15)    g1&r4g2.|:21@w1:|
167: (t15)    o4@49e1&e1&e1
168:
169: /        CHOIRUS-----------------------------------------
170: (t7)     i0@92o4@U90@v90q8@p6318@G12@e40,20[k.sign+f]
171: (t7)     |:16@w1:|'e1a1''f1b1'
172: (t7)     'e1<c1''f1<d1''>g+*384<e*384b*384'|:20@w1:|
173: (t7)     @74o5@v80l16defgab<cdefgab<cde
174: (t7)     o6>ab<cdefgab<cdefgab
175: (t7)     o5defgab<cdefgab<cde
176: (t7)     o6>ab<cdefgab<cdefgab
177: (t7)     |:12@w1:|
178: (t7)     |:o5>b<cdefgab<cdefgab<c
179: (t7)     o5efgab<cdefgab<cdef:|
180:
181: /        BRASS-------------------------------------------
182: (t13)    i0@62o4@U120@v120q8@p6318@G12@e45,30[k.sign+f]
183: (t13)    |:22@w1:|
184: (t13)    r2bab<cdcr>b&baga
185: (t13)    b4.e8&e2<d4.c8&c4r4l8r2>bab<cdcr>b&baga<c1r4f2.
```

```
186: (t13)    @63@e40,50r2>bab<cdcr>b4agab4.e&e2
187: (t13)    @62@e90,100<d4.c&c4r4
188: (t13)    @63@e40,50r2>bab<cdcr>b4aga<e1r4f2.|:4@w1:|
189: (t13)    @62@e90,100>de2..r1de&e2deg4.f&f2d1..l16defg
190: (t13)    a2.gab<cd1'g4.c4.>g4.''f*120>b*120f*120'r1
191: (t13)    'f4.>b4.f4.''g*120c*120>g*120'r1
192: (t13)    'g4.c4.>g4.''f*120>b*120f*120'r1'f4.>b4.f4.'
193: (t13)    'g4.c4.>g4.''a4.d4.>a4.''g4.c4.>g4.'r2
194: (t13)    |:4@w1:|
195:
196: /        BRASS & TRUMPET-------------------------------------
197: (t14)    i0@62o4@U120@v120q8@p6318@G12@e45,30[k.sign+f]
198: (t14)    |:16@w1:|>a1b1
199: (t14)    <c1d2.l16>abcde1r1
200: (t14)    o2l8<e2.{ega}4b1
201: (t14)    g2.{gab}4<d4.c&c>a<c>be2.{ega}4b1g1|:13@w1:|
202: (t14)    @57o4@v105'd>a''e2..>b2..'r1'd>a'
203: (t14)    'e*120>b*120''d>a''e>b'd4.c+&c+2r1r1
204: (t14)    @62r1r2l16defefgab@57l8<c4.>b&b2r1b4.<c&c2r1
205: (t14)    <c4.>b&b2r1b4.<c&c4d4.c4.r2
206: (t14)    |:4@w1:|
207:
208: /        MARIMBA-------------------------------------------
209: (t8)     i0@13o5@U90@v90q8@p6318@g12@e40,10[k.sign+f]
210: (t8)     'b<e'ee'b<d'ee'g<c'e
211: (t8)     e'a<d'ee'b<e'e'<cf!'e
212: (t8)     'b<e'ee'b<d'ee'g<c'e
213: (t8)     e'a<d'ee'b<e'e'fb'e
214: (t8)     'b<e'ee'b<d'ee'g<c'e
215: (t8)     e'a<d'ee'b<e'e'<cf!'e
216: (t8)     'b<e'ee'b<d'ee'g<c'e
217: (t8)     e'a<d'ee'g<c'e'f<d'e
218: (t8)     |:'b<e'ee'b<d'ee'g<c'e
219: (t8)     e'a<d'ee'b<e'e'<cf!'e
220: (t8)     'b<e'ee'b<d'ee'g<c'e
221: (t8)     e'a<d'ee'g<c'e'fb'e:||:22@w1:|
222: (t8)     'a<d'dd'g<c'dd'fb'dd'g<c'dd'a<d'd'g<c'd
223: (t8)     'a<d'dd'g<c'dd'fb'dd'g<c'dd'a<d'd'a<d'd
224: (t8)     |:4@w1:|addgddfddgddadgf
225: (t8)     addgddfddgr2.|:4@w1:|
226: (t8)     o4<c4.>b8&b2|:7@w1:|
227:
228: /        BASS----------------------------------------------
229: (t9)     i0@34o2@U110@v110q8@p63l16@M90@H24@S5,5
230: (t9)     @e0,0[k.sign+f]
231: (t9)     e1&e1eeeee2.l8r<c>bagab<c>e1e1
232: (t9)     l16eeeee2.e1l8|:3eeeeeeee:|e<c>bagab<c>
233: (t9)     |:4eeeeeeee:|aaaaaaaabbbbbbbb
234: (t9)     <cccccccc>dddddddde1r1|:3e4e4e4e4:|
235: (t9)     d4.c4>a<c>b|:7<e4e4e4e4:|d4.c4>a<c>b|:3<e4e4e4e4
:|
236: (t9)     eeeeafg4|:cddddddddddddddd:|
237: (t9)     de&e2.eeeeeeeede&e2deg4.f&f2cddddddddddddddd
238: (t9)     cdddddddddddl16defefgabl8<c4.>b&b2<dc>bbagaa+
239: (t9)     b4.<c&c2>edcdc>b<cc+
240: (t9)     <c4.>b&b2<dc>bbagaa+b4.<c&c4d4dc4.r2>|:4bbbbbbbb
:|
241:
242: /        CYMBAL--------------------------------------------
243: (t10)    @17o3@U110@v110l16@e30,30
244: (t10)    c#4r2.|:3r1:|c#4r2.r1r1
245: (t10)    @e20,30>a#4a#4a#4r4
246: (t10)    |:4a#4a#4a#4a#4:|
247: (t10)    <c#4>|:7a#4:|<c#4>|:5a#4:|r2|:4<c#4>g#4g#4g#4:|
248: (t10)    <c#4r2.r2.c#4>|:12a#4:|<c#4>a#4<c#4>a#4<c#4>
249: (t10)    |:7a#4:|<c#4c#4>a#4a#4<c#4c#4r2
250: (t10)    |:12>a#4:|r4.a#4r8<c#4c#4>|:7a#4:|
251: (t10)    <c#4>|:3a#4:|r4<|:4c#4:|>|:7a#4:|
252: (t10)    |:<c#4>|:7a#4:|:|a#8<c#8>r4a#4a#4
253: (t10)    <c#4>a#8<c#8r2c#4|:7>a#4:|
254: (t10)    <c#4>|:3a#4:|r4ra#8a#r2<c#4r8c#8r4c#4|:5c#4:|
255: (t10)    r8c#8r4c#4|:5c#4:|r8c#8r4c#4c#8r4.c#4c#4
256: (t10)    c#4c#8c#8r4c#4r2c#2c#4>|:7a#4:|<c#4>|:3a#4:|r1
257:
258: /        S.D-----------------------------------------------
259: (t11)    o2@U100l16
260: (t11)    r1r1ddddr2.r1r1r1
261: (t11)    ddddr2.r2.dddd|:4r4d4r4d4:|
262: (t11)    |:a#4d4r4d4|r4d4r4d4:|r4d4@U120<c32c32c>aaafff
263: (t11)    @U110|:4r4d4r4d4:|
264: (t11)    r4{v1ddv2ddv3dd}4{v4dv5dv6dv7dv8dv9d}4
265: (t11)    {v10dv11dv12dv13dv14dv15d}4|:d32<c32>af8:|
266: (t11)    d4d4|:15r4d4:|dddddddd
267: (t11)    |:6r4d4:|d32<c32>af8f8r4.d4|:6r4d4:|
268: (t11)    d32<c32>af8d4d4d4|:14r4d4:|
269: (t11)    r4d4r8d8d8d8|:6r4d4:|
270: (t11)    d32<c32>af8dr8.<c>af8dddd|:10r4d4:|
271: (t11)    r8<cc>aaffdd8.dd8.r4d4r4d4r8ddddddr2
272: (t11)    |:6r4d4:|d8afr8d8af8.dddd
273:
274: /        B.D-----------------------------------------------
275: (t12)    o2@U120@v120l16
276: (t12)    c4r2.r1r1r1c4r2.r1
277: (t12)    l8r1ccccccr4|:c4rcrcr4ccr4rcr4:|
278: (t12)    c4r4.cr4c8c8r4.cr4c4r4.cr4c8c8r2.|:4c4r4c8c8r4:|
279: (t12)    c4r2.r8.c16r4r16c16cr4|:6c4r4:|
280: (t12)    c4rcrcr4|:4c4r4:|c4a#4ccr4c4r2.
281: (t12)    |:6c4r4:|r8.c16rcrcr4|:4c4r4:|
282: (t12)    c4rcrc16c16r4r.c16r4.c16c16r4
283: (t12)    |:4c4r4ccr4:|ccr4ccr4c4r4ccr4ccr4ccr4c4rcr2
284: (t12)    |:3c4r4ccr4:|r8.c16r16cc16r.c16r4|:c4rcrcr4c4r4c
cr4:|
285: (t12)    c4rcrcr4c16c16rr4rc16c16rc16c16c8.c16rcrc16c16rc
286: (t12)    c4r4c4r4|:c4r4ccr4c4rcrcr4:|
287:
288: (t1)     [loop]
289: (t2)     [loop]
290: (t3)     [loop]
291: (t4)     [loop]
292: (t5)     [loop]
293: (t6)     [loop]
294: (t7)     [loop]
295: (t8)     [loop]
296: (t9)     [loop]
297: (t10)    [loop]
298: (t11)    [loop]
299: (t12)    [loop]
300: (t13)    [loop]
301: (t14)    [loop]
302: (t15)    [loop]
303:
304: (p)
```

リスト6　バキャロスの遺跡のカウンタ表示

```
 1:00000060 00002E80   2:00000060 00002E80   3:00000060 00002E80   4:00000060 00002E80
 5:00000060 00002E80   6:00000060 00002E80   7:00000060 00002E80   8:00000060 00002E80
 9:00000060 00002E80  10:00000060 00002E80  11:00000060 00002E80  12:00000060 00002E80
13:00000060 00002E80  14:00000060 00002E80  15:00000060 00002E80
```

リスト7　エンターテイナー


```
 1: (i)
 2: .comment Scott Joplin作曲『The Entertainer』 by kunkun
 3:
 4: (v1,0,58,15,2,0,180,1,1,1,1,3,0,
 5: 29,4,0,5,1,37,2,1,7,0,0,
 6: 22,9,1,2,1,42,2,12,0,0,0,
 7: 29,4,3,6,1,37,1,3,3,0,0,
 8: 15,7,0,4,10,0,2,1,0,0,1)
 9:
10: (m1,2048)(aFM1,1)
11: (m2,1792)(aFM2,2)
12: (m3,1792)(aFM3,3)
13: (m4,1792)(aFM4,4)
14: (m5,1536)(aFM5,5)
15: (m6,1536)(aFM6,6)
16: (m7,1536)(aFM7,7)
17: (m8,1536)(aFM8,8)
18:
19: (t1)
20: l16v13t90@1p2v12q6r64o5dec>a8bq5g8q6dec>a8bq5g8q6
21: dec>a8bag+q5g4p3q6<<g8r32v11dd+|:p3o4e<c8>e<c8>e<
22: c8&c4v16eff+gefg8df8e4.v11>dd+e<c8>e<c8>e<c8.&c4
23: v16cgca<c>f+8<d>f+af4.v11>dd+p3e<c8>e<c8>e<c8&c4
24: v16eff+gefg8df8v15e4.<cdecde8cdcecde8cdv16c>gefg8
25: dt100f8r64t90|1o5e4.v11>dt100d+r64t90:||2o5e4&ev16
26: >cdt100d+r64t90p3e8ae8cdd+e8ae8<ec>gab<cdedcd>g<ef
27: gagefe8ae8cdd+e8ae8gat105a+r64t90p3dd8c8af+t105dr64
28: t90>b4&bcdd+e8ae8cdd+e8ae8<ec>gab<cdedcdc4&c>gf+
29: t105gr64t90p3<c8>a<c8>a<c>ag<ceg8ect105>gr64t90p3
30: f+8f+8ff8t91e&e4.t300r64t90v11dp3d+e<c8>e<c8>e<
31: c8&c4v16eff+gefg8df8e4.v11>dd+e<c8>e<c8>e<c8.&c4
32: v16cgca<c>f+8<d>f+af4.v11>dt105d+r64t90p3e<c8>e<
33: c8>e<c8&c4v16eff+gefg8df8e4.v15<cdecde8cdcecde8cd
34: cv16>gefg8df8e4.t150r64r64t90v13p3r8>d8c+d8c+d8ra
35: fa<cdc>ae8d+e8d+e8r<c>g<cdedc>b8a+b8a+b8r<fafgagv16
36: fccc4c8c8>eee8t100e8r64t90v13d8c+d8c+d8rafa<cdc>a
37: e8d+e8d+e8r<c>g<cdedc>ag+aa8a8<ced+ea8<c>gv16ec8>
38: f+8fe8e&e8.t97r8.r64t90v11dp3d+e<c8>e<c8>e<c8&c4
39: v16eff+gefg8df8e4.v11>dd+e<c8>e<c8>e<c8.&c4v16cgc
40: a<c>f+8<d>f+af4.v11>dd+t300r64t90v11p3e<c8>e<c8>e
41: <c8&c4v16eff+gefg8df8e4.v15<cdecde8cdcecde8cdv16c
42: >gefg8df8e4.r64
43:
44: (t2)
45: l16v13@1p2v13q6r64o6dec>a8bq5g8q6dec>a8bq5g8q6dec>
46: a8bag+q5g4p3q6<b8r32v11r8|:p3rr8.r8.r8v16r4o6cdd+
47: ecde8>b<d8c4.v11r8.r8.r8.r8.v16r4>arf+r8<e8rcrd4.
48: v11r8p3rr8.r8.r8v16r4cdd+ecde8>b<d8v15c4.r4rr2r4
49: v16recde8>b<d8r64|1r64o6c4.v11r8:||2o6c4&cv16>>ef
50: f+r64p3g8rg8eff+g8rg8r1r8.<g8rg8eff+g8rg8r8.r64p3
51: gg8f+8r8.r64g4&g>eff+g8rg8eff+g8rg8r2r8.r2r64p2
52: r64<c8>a<c8>a<c>ag<ceg8ec>gp3a8<c8ed8c&c4.r64v11
```

```
 53: rp3r4rr8rr8r4v16<cdd+ecde8>b<d8c4.v11r8.r8.r8.r8.
 54: v16r4>arf+r8<e8rcrd4.v11r8r64p3rr8.r8.r8v16r4cdd+
 55: ecde8>b<d8c4.v15r4rr2r4rv16ecde8>b<d8c4.r64r64v13
 56: p3r8>>f8ef8ef8r8<dr8.r8>g8f+g8f+g8r8<er4rd8c+d8c+
 57: d8r4rr8v16r<ccc4c8>g8>ggg8g8r64v13f8ef8ef8r8<dr8
 58: r8.>g8f+g8f+g8r8<er2f8f8r4r4v16rr8c8ed8c&c8.r64r8.
 59: v11rp3r4rr8.r8v16r4<cdd+ecde8>b<d8c4.v11r8.r8.r8.
 60: r8.v16r4>arf+rr<e8rcrd4.v11r8r64v11p3rr8.r8.r8v16
 61: r4cdd+ecde8>b<d8c4.v15r4rr2r4v16recde8>b<d8c4.r64
 62:
 63: (t3)
 64: l16v13@1p2v13q6r64r8.r8.q5r4rr8.r4rr4rr4p3o5d8q6
 65: r32v11r8|:p2r64_lo4e<c8>e<c8>e<c8&c4v16eff+gefg8d
 66: f8e4.v11>dd+e<c8>e<c8>e<c8.&c4v16cgca<c>f+8<d>f+a
 67: f4.v11>dd+p2e<c8>e<c8>e<c8&c4v16eff+gefg8df8e4.
 68: v15<cdecde8cdcecde8cdv16c>gefg8df8|1v11o5e4.>dd+
 69: r64:||2r64o5e4&ev16>cdd+p2r64e8ae8cdd+e8ae8<ec>ga
 70: b<cdedcd>g<efgagefe8ae8cdd+e8ae8gaa+p3bb8b8r8.r64
 71: r64>b4&bcdd+e8ae8cdd+e8ae8<ec>gab<cdedcdc4&c>gf+
 72: gp3>f8f8f+8f+8g8g8g8g8r64p2r64<f+8f+8ff8e&e4.v11
 73: p2r64dd+e<c8>e<c8>e<c8&c4v16eff+gefg8df8e4.v11>d
 74: d+e<c8>e<c8>e<c8.&c4v16cgca<c>f+8<d>f+af4.v11>dd+
 75: p2r64e<c8>e<c8>e<c8&c4v16eff+gefg8df8v15e4.<cdecd
 76: e8cdcecde8cdv16c>gefg8df8e4.r64v13p2r64r8>d8c+d8
 77: c+d8rafa<cdc>ae8d+e8d+e8r<c>g<cdedc>b8a+b8a+b8r<f
 78: afgagv16fccc4c8c8>eee8e8v13r64d8c+d8c+d8rafa<cdc>
 79: ae8d+e8d+e8r<c>g<cdedc>ag+aa8a8<ced+ea8<c>gv16ec8
 80: >f+8fe8e&e8.r8.v11p2r64dd+e<c8>e<c8>e<c8&c4v16ef
 81: f+gefg8df8e4.v11>dd+e<c8>e<c8>e<c8.&c4v16cgca<c>
 82: f+8<d>f+af4.v11>dd+v11p2r64e<c8>e<c8>e<c8&c4v16ef
 83: f+gefg8df8v15e4.<cdecde8cdcecde8cdcv16>gefg8df8e4.
 84:
 85: (t4)
 86: l16v13@1p2v13q6r64r8.r8.q5r4rr8.r4rr4rr4p3o5g8q6
 87: r32v11r8|:p2r64_lrr8.r8.r8v16r4o6cdd+ecde8>b<d8
 88: c4.v11r8.r8.r8.r8.v16r4>arf+r8<e8rcrd4.v11r8p2r
 89: r8.r8.r8v16r4cdd+ecde8>b<d8c4.v15r8.r2r4r8v16recd
 90: e8>b<d8|1v11o6c4.r8r64:||2r64o6c4&cv16>>eff+p2r64
 91: g8rg8eff+g8rg8r4r2r4r8.<g8rg8eff+g8rg8rr8r64dd8c8
 92: af+dr64g4&g>eff+g8rg8eff+g8rg8rr2r8r2p3>a8a8a8a8<
 93: c8c8c8c8r64p2r64a8<c8ed8c&c4.v11p2r64r8.r8.r8.r8
 94: v16r4<cdd+ecde8>b<d8c4.v11r8.r8.r8.r8.v16r4>arf+
 95: r8<e8rcrd4.v11r8p2r64rr8.r8.r8v16r4cdd+ecde8>b<d8
 96: v15c4.rr1v16recde8>b<d8c4.r64v13p2r64r8>>f8ef8ef8
 97: r8<dr4r>g8f+g8f+g8r8<err4d8c+d8c+d8r4r8.v16r<ccc4
 98: c8>g8>ggg8g8v13r64f8ef8ef8r8<dr4r>g8f+g8f+g8r8<e
 99: r2f8f8r2v16rr8c8ed8c&c8.r8.v11p2r64r8.r8.r8.r8v16
100: r4<cdd+ecde8>b<d8c4.v11rr4rr8.r8.v16r4>arf+r8<e8r
101: crd4.v11r8v11p2r64rr8.r8.r8v16r4cdd+ecde8>b<d8v15
102: c4.r4rr2r4rv16ecde8>b<d8c4.
103:
104: (t5)
105: l8v13@1p3v13q6o5d16e16c16>ab16q5gq6d16e16c16>ab16
106: q5gq6d16e16c16>ab16a16g+16q5g4q6r64>gr32v11<<g|:p3
107: o3ce>g<g>f<f>ev16<e>g<e>g<fceev11gce>g<g>f<fv16>e
108: d+d<f+df+g>gav11bp3<ce>g<g>f<f>ev16<e>g<e>g<fceg
109: v15rcg>a+<g>a<a>g+v16<g+>g<g>g<gr64|1o3c>gcv11<g
110: r64:||2o3c>gcr64rp3c<g>g<gcg>g<g>f<afg+eg>g<gcg>g
111: <gcg>ed+r64d<gdar64g>fedc<g>g<gcg>g<g>f<afg+egca+
112: r64p3<cccceeeer64p3>ddggc>gcr64v11p3r<ce>g<g>f<f>
113: ev16<e>g<e>g<fceev11gce>g<g>f<f>ed+v16d<f+df+g>ga
114: br64v11p3<ce>g<g>f<f>ev16<e>g<e>g<fcegv15rcg>a+<g
115: >a<a>g+v16<g+>g<g>g<gc>gcr64r64v13p3rf<f>a<f>a<f>
116: a<fce>g<ece>g<e>g<f>b<f>g<fdfv16d+d+4d+er4.r64v13
117: >f<f>a<f>a<f>a<fce>g<ece>g<e>fdefg<gf+gv16a>dgb<c
118: >gcr64v11rp3<ce>g<g>f<f>ev16<e>g<e>g<fceev11gce>g
119: <g>f<f>ed+v16d<f+df+g>gabr64v11p3<ce>g<g>f<f>ev16
120: <e>g<e>g<fcegv15rcg>a+<g>a<a>g+v16<g+>g<g>g<gc>gcr64
121:
122: (t6)
123: l8v13@1p3v13q6o6d16e16c16>ab16q5gq6d16e16c16>ab16
124: q5gq6d16e16c16>ab16a16g+16q5g4q6r64>gr32v11<b|:p3
125: ro3gga+faev16grgrgrggv11brgga+fav16ed+darabgav11b
126: p3rgga+faev16grgrgrg<cv15rcc>a+<c>a<c>g+v16<c>g<c
127: r>br64|1o4c>gcv11br64:||2o4c>gcr64rp3c<crcrcrcrcr
128: crcrcrcrcrc>ed+r64dbr<cr64>bfedc<crcrcrcrcrcrcrc
129: r64p3ffd+d+r4rrr64p3c>abb<c>gcr64v11p3r4gga+fae
130: v16grgrgrggv11brgga+faed+v16darabgabr64v11p3rgga+
131: faev16grgrgrg<cv15rcc>a+<c>a<c>g+v16<c>g<cr>b<c>g
132: cr64r64v13p3r4arararargrgrgrgrgrgrgrgv16f+f+4f+g
133: r4.r64v13rarararargrgrgrgfdefg<cccv16d>dgb<c>gc
134: r64v11rp3rgga+faev16grgrgrggv11brgga+faed+v16dara
135: bgabr64v11p3rgga+faev16grgrgrg<cv15rcc>a+<c>a<c>
136: g+v16<c>g<cr>b<c>gcr64
137:
138: (t7)
139: l8v13@1p3v13q6r.r.q5r4r16r.r4r16r4r16r4r64rr32
140: v11r|:p3ro4cq6rcrcrv16crcr>br<ccv11r4crcrcv16r4rc
141: rcr4rv11rp3rcrcrcrv16crcr>br<cev15r4ererfrv16fre
142: r4r64|1r4rv11rr64:||2r4rr64rp3r2r4rr4o4frfrererer
143: erer4r64rdrdr64dr1r4rrfrfrerer64p1r64>fff+f+gggg
144: p3r2r4rr64v11p3r4<crcrcrv16crcr>br<ccv11r4crcrcr4
145: v16rcrcr2r64v11p3rcrcrcrv16crcr>br<cev15r4ererfr
146: v16frer2rr64r64v13p3r1r4crcrcrcr>brbrbrbv16<cc4cc
147: r4.r64v13r1rcrcrcrcr2red+ev16f+r2r4r64v11rp3rcrcr
148: crv16crcr>br<ccv11r4crcrcr4v16rcrcr2r64v11p3rcrcr
149: crv16crcr>br<cev15r4ererfrv16frer2rr64
150:
151: (t8)
152: l8v13@1p3v13q6r.r.q5r4r16r.r4r16r4r16r4r64rr32v11
153: r|:p1r64q6o3ce>g<g>f<f>ev16<e>g<e>g<fceev11gce>g<
154: g>f<fv16>ed+d<f+df+g>gv11abp1<ce>g<g>f<f>ev16<e>g
155: <e>g<fcegv15rcg>a+<g>a<a>g+v16<g+>g<g>g<g|1r64o3c
156: >gcv11<g:||2r64o3c>gcrp1r64c<g>g<gcg>g<g>f<afg+eg
157: >g<gcg>g<gcg>ed+r64d<gdar64g>fedc<g>g<gcg>g<g>f<a
158: fg+egca+p1r64aaaa<ccccr64p1>ddggc>gcv11rp1r64<ce>
159: g<g>f<f>ev16<e>g<e>g<fceev11gce>g<g>f<f>ed+v16d<f+
160: df+g>gabp1v11r64<ce>g<g>f<f>ev16<e>g<e>g<fcegrv15
161: cg>a+<g>a<a>g+v16<g+>g<g>g<gc>gcr64v13p1r64rf<f
162: >a<f>a<f>a<fce>g<ece>g<e>g<f>b<f>g<fdfv16d+d+4d+
163: er4.v13r64>f<f>a<f>a<f>a<fce>g<ece>g<e>fdefg<gf+
164: gv16a>dgb<c>gcv11rp1r64<ce>g<g>f<f>ev16<e>g<e>g<f
165: ceev11gce>g<g>f<f>ed+v16d<f+df+g>gabp1v11r64<ce>g
166: <g>f<f>ev16<e>g<e>g<fcegv15rcg>a+<g>a<a>g+v16<g+>g
167: <g>g<gc>gc
168:
169: (p)
```

リスト8　エンターテイナーのカウンタ表示

```
1:000025A1 00000000   2:000025A1 00000000   3:000025A1 00000000   4:000025A1 00000000
5:000025A1 00000000   6:000025A1 00000000   7:000025A1 00000000   8:000025A1 00000000
```

リスト9　中央競馬のファンファーレ1

```
 1: .Comment 中央競馬ファンファーレ(1) JRA日本中央競馬会
 2: /                                          copy by  Ysu
 3: (i)
 4: /-----------------------------------------------------
 5: /     AR DR1 DR2  RR  DL  TL  RS  ML DT1 DT2 AME  BRASS(tp)
 6: (@ 1,26,  2,  1,  5,  1, 26,  0,  1,  0,  0,  0
 7:       31,  4,  1,  7,  1,  0,  0,  1,  0,  0,  0
 8:       31,  4,  1,  7,  1,  2,  0,  2,  0,  0,  0
 9:       28,  6,  1,  7,  9,  0,  0,  1,  0,  0,  0
10: /     AL  FB  OM
11:        5,  7, 15)
12:
13: /     AR DR1 DR2  RR  DL  TL  RS  ML DT1 DT2 AME  BRASS(tb)
14: (@ 2,27,  4,  0,  4,  1, 26,  0,  1,  0,  0,  0
15:       27,  2,  0,  6,  1,  0,  0,  0,  0,  0,  0
16:       27,  2,  0,  6,  1,  2,  0,  1,  0,  0,  0
17:       26,  7,  0,  6,  8,  0,  0,  2,  0,  0,  0
18: /     AL  FB  OM
19:        5,  7, 15)
20:
21: 57=crsh0.pcm,v70
22: /-----------------------------------------------------
23: (b0)
24: (m1,4000)(a1,1)  /     桜花賞の馬券を買いに行った帰りに
25: (m2,4000)(a2,2)  /     思いついたネタです。
26: (m3,4000)(a3,3)  /     ちなみに、僕は(2)のファンファー
27: (m4,4000)(a4,4)  /     レよりも(1)のファンファーレのほ
28: (m5,4000)(a5,5)  /     うがすきですねぇ。
29: (m6,4000)(a6,6)  /     あとは本馬場入場のテーマとか..
30: (m7,4000)(a7,7)  /     しかしどれだけの人がこの曲(!)を
31: (m8,4000)(a8,8)  /     知っていてくれるのだろうか。
32: (m9,4000)(a9,9)
33: /-----------------------------------------------------
34: (o130)
35: (t1)@ 1v15o5@k0,-5r|:3q6e-8e-16e-16q8e-8e8:|q7{e-e-e-eee}2
36: (t1)|:3q6f8f16f16q8f8g-8:|
37: (t1)q7{fffg-g-g-}2|:3g8g16g16g8a-8|g4r4:|r64{gggaaa}1r 8b1&b1
38: (t2)@ 1v15o5@k0,+5r|:3q6e-8e-16e-16q8e-8e8:|q7{e-e-e-eee}2
39: (t2)|:3q6f8f16f16q8f8g-8:|q7{fffg-g-g-}2|:3q6g8g16g16q8g8a-8
40: (t2)|g4r4:|q7r64{gggaaa}1r16b1&b1
41: (t3)@2@k0,-5ro3v13r1.e-4e4f4r4 r1f4g-4g4r4
42: (t3)|:r8g16a16b16a16g16r16r2:|
43: (t4)@2@k0,+5ro3v13r1.e-4e4f4r4 r1f4g-4g4r4
44: (t4)|:r8g16a16b16a16g16r16r2:|
45: (t5)@1@k0,+5ro3v13r1.e-4e4f4r4 r1f4g-4g4r4<v10
46: (t5)|:r8g16a16b16a16g16r16r2:|
47: (t6)@ 1q6v14o4@k0,-5r|:3q6e-8e-16e-16q8e-8e8:|
48: (t6)q7{e-e-e-eee}2|:3q6f8f16f16q8f8g-8:|q7{fffg-g-g-}2
49: (t6)|:3q6g8g16g16q8g8a-8|g4r4:|q7r64{gggaaa}1r16b1&b1
50: (t7)@ 1q6v13o3@k0,+5r|:3q6e-8e-16e-16q8e-8e8:|
51: (t7)q7{e-e-e-eee}2|:3q6f8f16f16q8f8g-8:|q7{fffg-g-g-}2
52: (t7)|:3q6g8g16g16q8g8a-8|g4r4:|q7r64{gggaaa}1r16b1&b1
53: (t8)@ 1q6v10o2@k0,+5r|:3q6e-8e-16e-16q8e-8e8:|
54: (t8)q7{e-e-e-eee}2|:3q6f8f16f16q8f8g-8:|q7{fffg-g-g-}2
55: (t8)|:3q6g8g16g16q8g8a-8|g4r4:|q7r64{gggaaa}1r16b1&b1
56: (t9)o3 ra2r1aaa2r1r2a2r2ar2rarr1r64r16a1,
57: (p)
```

▶プログラムのラベル名に時間を費やす自分に気がついた。　須田　周作(22)岡山県

リスト10　中央競馬のファンファーレ１のカウンタ表示

```
1:0000076B 00000000   2:0000075F 00000000   3:00000510 00000000   4:00000510 00000000
5:00000510 00000000   6:0000075F 00000000   7:0000075F 00000000   8:0000075F 00000000
9:000006FF 00000000
```

リスト11　中央競馬のファンファーレ2

```
 1: .Comment  中央競馬ファンファーレ(2) JRA日本中央競馬会
 2: /                                  copy by  Ysu
 3: (i)
 4: /-------------------------------------------------------
 5: /     AR DR1 DR2  RR  DL  TL  RS  ML DT1 DT2 AME  BRASS(tp)
 6: (@ 1,24,  2,  1,  5,  1, 26,  0,  1,  0,  0,  0
 7:      31,  4,  1,  7,  1,  0,  0,  1,  0,  0,  0
 8:      31,  4,  1,  7,  1,  2,  0,  2,  0,  0,  0
 9:      28,  6,  1,  7,  9,  0,  0,  1,  0,  0,  0
10: /     AL  FB  OM
11:        5,  7, 15)
12:
13: /     AR DR1 DR2  RR  DL  TL  RS  ML DT1 DT2 AME  BRASS(tb)
14: (@ 2,28,  4,  0,  4,  1, 26,  0,  1,  0,  0,  0
15:      27,  2,  0,  6,  1,  0,  0,  0,  0,  0,  0
16:      27,  2,  0,  6,  1,  2,  0,  1,  0,  0,  0
17:      26,  7,  0,  6,  8,  0,  0,  2,  0,  0,  0
18: /     AL  FB  OM
19:        5,  7, 15)
20:
21: /     AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME  SNARE
22: (@ 3,31, 18,  1,  7,  1,  0,  0,  3,  0,  1,  0
23:      31, 15, 15,  9, 15,  1,  0,  1,  0,  0,  0
24:      31, 20, 10,  7, 15,  5,  0,  1,  0,  1,  0
25:      31, 15, 15,  9, 15,  0,  0,  1,  0,  0,  0
26: /     AL  FB  OM PAN
27:        4,  7, 15,  3)
28:
29: 57=crsh0.pcm,v70
30: /-------------------------------------------------------
31: (b0)
32: (m1,4000)(a1,1)
33: (m2,4000)(a2,2)
34: (m3,4000)(a3,3)
35: (m4,4000)(a4,4)
36: (m5,4000)(a5,5)
37: (m6,4000)(a6,6)
38: (m7,4000)(a7,7)
39: (m8,4000)(a8,8)
40: (m9,4000)(a9,9)
41: /-------------------------------------------------------
42: (o100)
43: (t1)@ 1v14o4@k0,-2
44: (t1)|:q7b-4.b-16b-16b-8b-8b-8b-8|<f1>:|<g-2.
45: (t1)>b-8.b-16<{gfe-ce-f}2q8gr32gr8a1&a4
46: (t2)@ 1v14o4@k0,+2
47: (t2)|:q7b-4.b-16b-16b-8b-8b-8b-8|<f1>:|<g-2.
48: (t2)>b-8.b-16<{gfe-ce-f}2q8gr32gr8a1&a4
49: (t3)@ 1v12o3@k0,-2
50: (t3)|:q7b-4.b-16b-16b-8b-8b-8b-8|<f1>:|<g-2.
51: (t3)>b-8.b-16<{gfe-ce-f}2q8gr32gr8a1&a4
52: (t4)@2o3v13r1r4{b-b-b-}4b-8b-8r4 r1r4{g-g-g-}4g-8g-8
53: (t4)r4g2g4r32g4r8a1
54: (t5)@2o4@k0,-5v14r1r4{b-b-b-}4b-8b-8r4
55: (t5)r1r4<{g-g-g-}4g-8g-8> r4g2g4r32g4r8a1
56: (t6)@2o3@k0,+5v14r1r4{b-b-b-}4b-8b-8r4
57: (t6)r1r4<{g-g-g-}4g-8g-8> r4g2g4r32g4r8a1
58: (t7)@3@k256,-512o3v13l8r1r4{eee}4eer4 r1r4{eee}4ee
59: (t7)l8r4er4.err32err.v10|:32e64:||:16e58:||:8e48:|
60: (t8)@3@k256,-512o3v15l8r1r4{ccc}4ccr4 r1r4{ccc}4cc
61: (t8)l8r4cr4.crr32crr.v11|:32c64:||:16c58:||:8c48:|
62: (t9)o3r1r4a4a8a8r4
63: (t9)r1r4a4a8a8 r4a2a4r32a4r8a1
64: (p)
```

リスト12　中央競馬のファンファーレ2のカウンタ表示

```
1:000004CE 00000000   2:000004CE 00000000   3:000004CE 00000000   4:0000049E 00000000
5:0000049E 00000000   6:0000049E 00000000   7:0000049A 00000000   8:0000049A 00000000
9:0000049E 00000000
```

リスト13　ベンザエースのCMソング


```
 1: /-------------------------------------------------------
 2: /
 3: /                        1994/4/10
 4: /
 5: /                 ＣＭソング～ベンザエース～
 6: /
 7: /                   歌／鷲尾いさ子＆女の子
 8: /
 9: /                   programmed  by K.Ohata
10: /
11: /-------------------------------------------------------
12:
13: .comment ＣＭソング～ベンザエース～
14:
15: (i)
16: (b1)
17: (o112)
18:
19: (m1,4000)(amidi1,1)
20: (m2,4000)(amidi2,2)
21: (m3,4000)(amidi3,3)
22: (m4,4000)(amidi4,4)
23: (m5,4000)(amidi5,5)
24: (m6,4000)(amidi6,6)
25: (m7,4000)(amidi7,7)
26:
27: .sc55_reverb $10={7,3,3,95,80,60,80}
28:
29: /-------------------------------------------- ﾌﾙｰﾄ
30: (t1) r4n1@74v12q8l4o4@i$41,$10,$42
31: (t1) [k.sign -e,-b] b2<{fba}{fab}&b2{baf}{af>b}&b2.r
32: (t1) rrr<{fd&d}>b2r<{fd&d}>b2r<f4d2r
33: (t1) {'fa'&'fa''fd'} v8 cc2>b1
34:
35: /-------------------------------------------- ﾋﾞｵﾗ1
36: (t2) r4n2@42v8q3l4o5 @i$41,$10,$42
37: (t2) [k.sign -e,-b] 'fb<d''fb<d''gb<e''gb<e''b<df''b<df'
38: (t2) 'gb<e''gb<e' v6 |:4 'fb<d' :||:2 |:3 'fb<d' :| r :|
39: (t2) |:3 'b<df' :| r |:3 'b<df' :| q8
40: (t2) <{'af'&'af''fd'}> 'egb''d2fa''d1fb>b'
41:
42: /-------------------------------------------- ﾋﾞｵﾗ2
43: (t3) r4n3@42v9q5l4o4 @i$41,$10,$42
44: (t3) [k.sign -e,-b] 'fb<d''fb<d''gb<e''gb<e''b<df''b<df'
45: (t3) 'gb<e''gb<e' v7 |:4 'fb<d' :||:2 |:3 'fb<d' :| r :|
46: (t3) |:3 'b<df' :| r |:3 'b<df' :| q8
47: (t3) <{'af'&'af''fd'}> 'egb''d2fa''d1fb>b'
48:
49: /-------------------------------------------- ｾﾚｽﾀ
50: (t4) r4n4@ 9v8q4l4o6 @i$41,$10,$42
51: (t4) [k.sign -e,-b] bb<ccd>b<cc>brrrr1r1r1r1 q8
52: (t4) <{d&d>b} <cc>b1
53:
54: /-------------------------------------------- ｺﾝﾄﾗﾊﾞｽ
55: (t5) r4n5@44v 8q4l4o2 @i$41,$10,$42
56: (t5) [k.sign -e,-b] bb<ccd>b<cc>brrr |:2 bbbr :| <
57: (t5) |:2 dddr :| q8 {d&d>b}<cc>b1
58:
59: /-------------------------------------------- ｸﾗﾘﾈｯﾄ1
60: (t6) r4n6@72v12q6l4o5p1 @i$41,$10,$42
61: (t6) [k.sign -e,-b] r1r1r1{ddd}{dc>b}<{ddd}&d8r8r1
62: (t6) {fff}{fed}{fff}&f8r8r1 q8 {g&gg}aab1
63:
64: /-------------------------------------------- ｸﾗﾘﾈｯﾄ2
65: (t7) r4n7@72v12q6l4o5p2 @i$41,$10,$42
66: (t7) [k.sign -e,-b] r1r1r1r1{ddd}{dc>b}<{ddd}&d8r8
67: (t7) r1{fff}{fed}{fff}&f8r8 q8 {g&gg}aab1
68:
69: (p)
```

リスト14　ベンザエースのCMソングのカウンタ表示

```
1:000006C0 00000000   2:000006C0 00000000   3:000006C0 00000000   4:000006C0 00000000
5:000006C0 00000000   6:000006C0 00000000   7:000006C0 00000000
```

▶メモリとミニディスク，どちらを買うか。それが問題だ。　　佐藤　真(23)愛知県

◆宇宙刑事ギャバン

トランペットはもっと歯切れよくしたいところですが，出来は悪くありません。ただエフェクタでベースの定位がふらついているのがちょっと気になります。ベースのみ，レベルを半分くらいに落としてもいいかも。

◆闘え！ ダダンダーン

このチープさは，狙ったものだとすればウマイ。ギターの演奏がちょっとメリハリないですね。まあこれは凝り出すときりがないし，相当難しい技術になってしまいますが。

リズムのパンを楽器ごとに変更する方法なら，先月号と今月号のこのコーナーを参考にしてください。

◆バキャロスの遺跡

最近のコナミは非常に印象が薄いのですが，曲は相変わらずいいんですよね……。

曲は無限ループなので，変更しないパラメータは［do］～［loop］の外に出しましょう。そうしないと遅い楽器はループポイントでモタります。

パート間でのバランスのツメがちょっと甘く感じます。改善の余地はあるでしょう。

◆エンターテイナー

FM音源のちょっと弱いピアノが逆に功を奏して，なかなかいい味です。ただ，やはりベロシティをいじらないと表情がつきません。ピアノを窮めんとするならそのあたりもぜひ挑戦してはいかがでしょうか。

このリストはすぐにMIDI対応にできますから，そちらも楽しめるかもね。

◆JRAファンファーレ

ごめんなさい……。競馬はまったく見ないのでネタを知りません(これでも府中に住んでいるのだが)。

音色がちょっと苦しいです。内蔵音源でうまくまとめるのは難しいことなのですが，ゲーム音楽などはよい研究材料になりますよ。X68000には優れたゲームがたくさんありますしね。

◆ベンザエースのCMソング

耳コピ初挑戦だそうですね。こういう，身近で扱いやすい題材からスタートするのが上達への早道ですね。

それでは6月号の続きです。6月号のリストをご覧になりながらお読みください。

各楽器のグローバルな設定は前半部分の共通コマンド(ピリオドで始まる)で行っています。

.roland_exclusive DEV,MDL ｛アドレス，……｝

DEVはデバイスID，MDLはモデルIDです。モデルIDは楽器の種類を表し，デバイスIDは同じ楽器が複数ある場合に，それらを区別するために用います。あまり意味がわからない人はリストの値をそのまま使うといいでしょう。

SC-55のある人は，パネルのALLボタンを押してみてください。MIDI CHの表示が17になれば，17-1=16=$10がデバイスIDです。

では，主なパラメータについて簡単に解説しておきます(CM,SC重複は勘弁)。

●master tune

これは音源全体のピッチを微妙に調節したい

▼ (進)の「ちょっといいですかぁ?」 ▲

場合に設定します。通常はデフォルトのパラメータでよいでしょう。

●master volume

音源全体で音声の出力レベルを調節できます。パートで設定できる”V”や”U”とは違います。

ミキサーのセッティングは固定しておいて，こちらでミキシングバランスを取れるので便利でしょう。

●master key shift

全体を移調します。SC-55のみですが，これはたいていパートでも設定できるのであまり必要ないですね。

●ptl (voice) reserve

パーシャルリザーブです。SC-55の場合は，音源マニュアルにもあるように，先頭のパラメータがパート10のリザーブを行います。内部ではこのようにパートが不規則に並んでいるわけですが，これはトラック上からパートに対してメッセージを送る際にも関係してきますので，要注意です。

●エフェクタ関連

CM-64はあまりにも簡単なので省略して，SC-55での知っておきたいポイント。

pre-lpfというのはローパスフィルタのことで，数値が大きいほどエフェクト音が徐々に「こもって」いきます。

feedbackは出力を入力へ返す度合いを設定するものですが，「reverb delay feedback」はディレイに対するものなので，あまり使わないでしょう。一方「chorus feedback」はコーラスすべてに有効なフィードバックで，こちらは普通に使います。ただしほどほどに。

それぞれ最後のパラメータは，「リバーブ音をコーラスへ」「コーラス音をリバーブへ」送るレベルを調節するものですが，これが大きくなると発振してパニック状態になります。オーディオ装置や耳を壊さないように注意。

ところで，投稿作品を聴いているとどうもエフェクタのレベルが高すぎる傾向にあるように思います。エフェクタをかけないSC-55の音なんかは結構不満タラタラなのはわかるのですが，かけ過ぎは音像がぼやけてキレがなくなるので，注意したいところです。

よりよいエフェクトを目指し，まずはプロの使い方を勉強してみるのがいいと思います。さまざまな音楽ソースを，ヘッドフォンを使い，耳を澄ませて聴いてみてください。

◆　◆　◆

ではMMLの解説です。6月号のリストではトラックヘッダ(tn)をかなり省略してあります。省サイズやtコマンドの検索に貢献するものです。もうご存じの方も多いでしょう(ただしこのワザを使った場合はコンパイラの都合で，行の先頭にポルタメントがあるとそれをトラックヘッダと誤認してしまい，エラーが出てしまいます)。

6月号のパート割り計画ではチャンネル1～4を使うことにしたので，リストの先頭でそのようなトラックアサインを行います。ドラムパートのみシンバル系を独立させて，2トラックで作ります。

で，ローランド系楽器を使う場合はトラックヘッダを書いてすぐに，@iコマンドで楽器の種別を定義するクセをつけるようにします。これはコマンド送信の際に内部で使用するものですから，必ず書くように。

・トラック1 ベース(LA1，チャンネル1)

まずはy126,1でモノモードに。そしてパッチエリアをちょこっといじってリバーブをカットしています。

・トラック2 コード(PCM1，チャンネル2)

エクスプレッション(音量だがボリュームとは区別されるので，とても有効)というコントロールチェンジに対し，拡張ARCCを使っています。要はソフトウェアエンベロープ。

拡張ARCCは，任意のコントロールチェンジへいろんな波形情報を送信できるので，表現力アップにつながります。コントロールチェンジにもいろいろありますが，音源のマニュアルを見てどれが使えるかを知っておくのもいいでしょう。

・トラック3 メロディ(PART1，チャンネル3)

カットオフを拡張ARCCで操作して，効果音的な表現を行っています。対象コントロールチェンジはデータ・エントリー。あらかじめ，NRPNでデータ・エントリーに書いた数値が何に対してのパラメータなのかを指定してあります。

また，音色のアタックとリリースを変えてあります。NRPN送信コマンドの使用例として参考にしてください。

・トラック4，5 ドラム(PART2，チャンネル4)

NRPN送信コマンドにより，ドラムパート各楽器のピッチ，音量，リバーブのかかり具合い，パンポットを変更しています。リズム隊のセッティングですね。

あとはベロシティシーケンスで流れに味つけをしてあります。たいしたモノではありませんが……。

リスト内にも注意書きがあるように，ベロシティシーケンスはコンパイル時に全部，展開されて埋め込まれていきます。よって，

z100,50 |:16 c :|

と，

z100,50 |:8 cc :|

とは別の演奏になります。繰り返し記号がない状態でベロシティが決められると考えればいいでしょう。

◆　◆　◆

私信で恐縮なのですが，「X1での投稿が聴けなかったよ～ん」と以前書いたら，ご丁寧にもテープに録音して送ってくださった山田美保さん，ありがとうございました。ご自分で制作されたというほかの曲も聴かせてもらいました。いやあ正直いって，女性でこういった趣味をお持ちの方はあまり見かけないので，曲ともども感激しました。男性にはないセンスって，絶対ありますね。　(進藤慶到)

(善)のゲームミュージックでバビンチョ

西川善司

この原稿は，茨城県の水戸で書いている。ここで暮らして数週間経つが，ゲームセンターの格闘ゲームの乱入台の人気がいまひとつなのが印象的だ。しかし，たまにやっている人を見つけて乱入し，2～3戦こっちが勝ったりしたら大変。台を蹴ったり灰皿投げつけたり，1度は本当のストリートファイトになったこともあり。しかも，こういうのは1度ならず何度も。ゲームセンターという，一見，郷土に無関係なものから風土の違いを感じている今日この頃だ。

さて先月は，私の多忙&編集の連休進行，そしてサンプルテープの入手の遅れと，不運が積み重なり，久々に落ちてしまった。というわけで，今月の紹介分はすでに発売中のものもある。了承されたし。

●ナムコ・ゲームサウンド・エクスプレス Vol.12
「ギャラクシアン³・プロジェクトドラグーン・シアター6」
CD:VICL15026　1,500円(税込)
ビクターエンタテインメント　発売中

花博やワンダーエッグで話題になった，ナムコの最大26人同時プレイ可能な大型3D体感シューティングゲームが，6人同時プレイのコンパクトサイズになって僕らの町へやってきた。これはその「ギャラクシアン³・シアター6(GT-6)」のサントラアルバム+α。BGMはいわゆる情景描写系の曲。緊張感あふれるコードワークとブレークを多用したトリッキーなリズムがゲームプレイ時の興奮を蘇らせてくれる。トラック後半にはビデオ「スターブレード」に収録されていたゴージャスアレンジ版も全曲収録。さらにGT-6の日本語，英語両バージョンのプレイ再現シナリオも収録されている。今回は「スターブレード」のときみたいなおふざけはなし。

お勧め度　9

●イース・ピアノコレクション2
CD:KICA-1142　3,000円(税込)
キングレコード　発売中

以前，瀧氏が「CREATIVE COMPUTER MUSIC」連載時に絶賛していたCD「プレプリマ」シリーズのアレンジャー藤澤道雄による，完全ピアノソロの「イース」を題材にしたオールアレンジバージョンアルバム。「プレプリマ」同様，とにかくアレンジは大胆。主題を独特の変奏(変装!?)で綴り，原曲のモチーフを最大限に活用したピアノならではのパフォーマンス。なかなかの迫力だ。原曲のイメージとはかなり違うので賛否両論だろうが，私は気に入った。読書，勉強のBGMのレパートリーにも加えたい。

お勧め度　8

●ステレオドラマ・ツインビーPARADISE Vol.6
CD:KICA-7638　2,800円(税込)
キングレコード　6/22発売

昨年10月から文化放送でオンエアされていたツインビーのラジオドラマのCD化も，これで6枚め。今回は第21話から最終話(第24話)までを完全収録。突如現れた謎の巨大隕石に，地球は絶体絶命の危機に直面。今回ばかりはいつものワルモン博士のメカとのほのぼのバトルというわけにはいかないぞ。さぁどーするツインビーたち。明かされるツインビー出生の謎。ツインビーのパイロットである主人公ライト少年とツインビーの別れ。そしてワルモン博士の意外な行動に驚け。物語は急激にシリアスに進展する。ツインビーファンは必聴だ。

お勧め度　8

●Maria Ano/Samba for 'T'
CD:MSCD-1001　2,500円(税込)
MUSICIAN'S SQUARE　発売中

パソコン通信ネットワークPC-VANにはコンピュータ音楽SIG「ミュージシャンズスクエア(JSMU)」というのがあり，全国のアマチュアミュージシャンの発表の場になっている。そこで発表される曲のなかにはプロ顔負けの作品もあって，同人CDとして発売されている。で，そんなスーパーアマチュアミュージシャンのなかにはソロアルバムまで出すほどの実力者もいる。今回紹介するのは，そのひとり「あのまりあ」氏のソロアルバムだ。

きっかけは，ある日私が手に入れた「Bamboo Shuffle」「Mega-Pulse」の2曲のGS音源用データ。これらを聴いて，私はショックと深い感動を受けた。こんなすごい曲を作る人のソロアルバムならば，聴かなくては一生の不覚と，手に入れたのが今回のCDなのだ。上記の2曲はフュージョン系の普通の主題(いわゆるフックライン)のある音楽だったが，ソロアルバムのほうはジャズだった。それもラテン系の軽快なヤツだ。楽器もピアノ，ギターなどのアコースティック系がメイン。どの曲も即興性が強く，さわやかな進行と展開は「夏」と「地中海」をイメージさせる。

お勧め度　10

このCDの購入方法
・2,500円(税込)を郵便振替でPC-VANミュージシャンズスクエアに送金する。
・郵便振替口座番号は「東京5-360270」
・加入者名は「ミュージシャンズスクエア事務局」
・振替用紙の通信欄にCDタイトルを明記

なお，Oh!X編集部への問い合わせは御遠慮ください。

おわりに

カシオペアの向谷実のソロアルバム「Tickle the Ivory」(CD:ALCA-524)を聴いた。トラック11の「A Day in the Stars」がいい。陶酔して弾いている向谷実の様子が目に浮かぶ。

では，また来月。

ギャラクシアン³・プロジェクトドラグーン・シアター6

イース・ピアノコレクション2

連載 DōGA CGアニメーション講座 ver.2.50……………………………(第16回)

CGA入門キット「GENIE」(その1)

プロジェクトチームDōGA
かまた ゆたか

いま，これを立ち読みしている諸君！ とりあえず今月号は買っておこう。付録ディスクはコピーで入手してもいいが，本誌のカラーページがないと使いにくいぞ。さあ，今回は「GENIE」の使い方(基礎編)だ！

[1] GENIEとは

プロジェクトチームDōGAが，2年ぶりに付録ディスクを制作しました。CGA入門キット「GENIE」です。

「GENIE」は，これからCGAを始めてみよう，あるいはCGAを始めたがすぐ挫折してしまった，などという初心者の方々に，本格的なオリジナルCGAを手軽に制作・体験してもらうことを目的に開発されました。

「GENIE」(ジニー)という名は，ディズニー映画「アラジン」に出てくるランプの精の名前です。そう，あの3つの願いをかなえてくれるという魔法使いです。この「GENIE」も，CGの初心者のために，3つの願いをかなえてくれるでしょう。

まず1つは，オリジナルの宇宙戦闘機や宇宙戦艦をデザインする機能。2つ目は，それらのオリジナルメカに動きを設定する機能。そして3つ目は，作画計算を行い，実際にCGAを制作する機能です。

「GENIE」は，メニューシステムと，DōGA CGAシステムの一部と，データベースから構成されています。

このデータベースが「GENIE」の大きな特徴といえます。データベースには，宇宙戦闘機や宇宙戦艦に使えそうな形状のパーツが約200(FD版の場合，約50)収められています。

ですから皆さんは，オリジナルメカをいちから作るのではなく，いろんなパーツを組み合わせてデザインすることができます。ちょうど，CG版のプラモデルといったイメージです。

CGでは，個々のパーツをただくっつけるだけでなく，大きさを変えたり，食い込ませたり，本物のプラモデルにはない柔軟性がありますので，200ものパーツがあれば，あなたの創造力次第でさまざまなデザインが可能です。

この方式ですと，CAD.Xを使いこなせなかった方でも，10分もあれば，ものすごく複雑なメカがサクサクとできます。実は，第6回CGAコンテストのオープニングの宇宙艦隊も「GENIE」で制作しているのです。あの宇宙船が，どれも5〜10分で制作されていたとは思わなかったでしょう。

この「GENIE」を使えば，誰でもあの程度のCGAが作れるようになります。それではさっそく，皆さんもがんばって挑戦してみましょう。

[2] 付録ディスクの展開

当然のことながら付録ディスクは圧縮されていますので，まず展開しなければいけません。この展開の作業は，簡単ですがかなり時間がかかります。ですから，とりあえず展開を始めて，その間にゆっくり本誌を読むのがよいでしょう。

1) HD版とFD版の違い

「GENIE」は，HD(ハードディスク)にインストールして使用する場合と，FD(フロッピーディスク)だけで使用する場合では，若干の違いがあります。

FD版は，当然のことながら容量の制限が厳しいため，使用できるパーツの数が約4分の1になってしまいます。14〜17ページの「GENIEのパーツ一覧表」の写真のなかで，左下に「FD」と書かれたものが，FD版でも使用可能なパーツです。

また，FD版では，ドライブ1のディスクをワークとして使用しますが，ワークディスク1枚に収まらないような長いアニメーションを制作することはできません。

▶CGAコンテスト事務局より◀

第6回CGAコンテストのビデオの申し込み，ありがとうございました。今年のビデオの発送作業は無事終了いたしました。正確にいうと，この原稿を書いている時点では，締め切りまぎわに申し込まれた方の分を準備している段階なので，何事もなければこの号が発売される頃には，すべて終了しているはずです。

ということで，申し込んだのにまだ届いていないという方は，控えのコピーを同封のうえ，DōGAまでお問い合わせください。

なお，今回は通信欄に，多くの励ましのお便り，要望，質問などを書いていただきました。特に「500円の攻防」については，多くの方が意見を述べておられました。これについては，ほかの質問などと合わせて，近く誌上で紹介したいと思います。

皆さん，CGAコンテストの経費について，大変心配されておられましたが，幸い第6回に関しては，ビデオの制作費が予想を下回ったこと，そして何より，定価3,000円を上回る金額を振り込んでくださった方が非常に多かったため，赤字はほとんど解消されました。この場を借りて，お礼申し上げます。

DōGA

つまり「GENIE」は，基本的にHDで使用することを前提にしているといえます。FDでは，一応，ひと通りのことができるといった程度で，いろいろと不都合があるでしょう。本格的にCGAを始めるには，HDは必須といえますので，FD版で不満が出るほどCGAにのめり込んでしまった人は，この際，HDの購入を検討されてはいかがでしょうか。

2) HDへのインストール

HDへインストールするためには，自分のHDのドライブ構成などをちゃんと理解できる程度の知識が最低限，必要です。また，ごくわずかとはいえ，ほかのファイルなどが破壊される可能性もありますので，そのへんの覚悟も必要でしょう。

HDへインストールするときは，付録ディスクを入れる前に，通常通りHDから起動しておきます。付録ディスクから起動してインストールすると，インストール後，「GENIE」がちゃんと実行できません。また，「GENIE」を実行するためには，容量が最低4Mバイト程度残っていなければいけませんので，必要に応じて整理してください。

準備ができましたら，ドライブ0に付録ディスクを入れ,そのドライブへと移ります。付録ディスクの中に「INSTALL.BAT」がありますから，

INSTALL⏎

で実行してください。すると，インストーラのメニュー画面が表示されます(写真1)。

そこで，「ハードディスクにインストールする」のボタンをクリックします。

するとウィンドウが2つ現れます。1つには，現在のドライブ構成と残り容量が表示されます。インストール先を決める参考にしてください。もう1つには，

「インストールするディレクトリをドライブ込みで入力して下さい」

と表示されますので，キーボードで，ディレクトリの名前を入力してください。ただ，このインストーラがKo-WINDOW上で動いている関係で，マウスカーソルがこのウィンドウのディレクトリ名を入力する枠の中にないと，キー入力を受けつけません。キーボードを叩いても何も文字が出ないときは，マウスカーソルを移動させてください。

また，このディレクトリ名はフルパスで入力してください。たとえば，「A:¥GENIE」はよいのですが，「A:GENIE」だと，「ドライブの指定が正しくありません」と文句をいってきます。

ディレクトリ名を確認すると，あとは自動的に展開コピーが開始されます。データベースのところは特にファイル数が多いので，相当待たされるでしょう。展開の様子は，ウィンドウに表示されます。エラーメッセージなどが出ていなければ無事展開が終了しています。

インストーラには,「サンプルデータフロッピーを作成する」というメニューがありましたが，HDへ展開する場合は無条件にサンプルデータもHDへインストールされますので，心配する必要はありません。

インストールが終了しましたら，そのディレクトリに移り，

GENIE⏎

で「GENIE」が起動します。ただ，command.xにパスが通っていないといけません。パスの通し方がわからなければ，「GENIE」をインストールしたディレクトリのBINの中に，command.xをコピーしておいても結構です。

CGA入門キット GENIE
ハードディスクにインストールする
システムフロッピーを作成する
サンプルデータフロッピーを作成する
インストーラを終了する

写真1 インストーラのメニュー画面

データの解説

[パーツの分類]

戦闘機：典型的な宇宙戦闘機のボディの形状
戦艦：典型的な戦艦のボディの形状
核：比較的単純で，いろいろなものをくっつける核となる形状
円盤：丸く，平べったい形状
棒：前後方向に細長い形状
平坦：平べったい形状
横腹：左右非対象で，ボディの左右につけるような形状
翼：ハネ。ジョイントとしても使える
エンジン：戦闘機，戦艦のエンジン，ノズルなど
タンク：燃料タンク，貨物的な形状
操縦席：ボディにくっつければ，そこがコックピット
艦橋：ボディに加え，戦艦の艦橋にするパーツ
楯：シールド。戦闘機のボディにもなりそう
砲台：レーザー砲，ミサイル砲など
連結：ジョイント。パーツとパーツを接続するパーツ
細部：ちょっともの足りないとき，ディテールをつけ加える
その他：分類しにくい，わけのわからない形状
基本：角錘，直方体といった単純な形状

[サンプル形状]

FI_A：スペースシャトル型の典型的な戦闘機
FI_B：ウイングの一部にシールドを使用した戦闘機
FI_C：ちょっと変わったデザインの戦闘機
BA_A：戦艦のボディを3つ並べたミサイル艦
BA_B：円盤のボディを前後逆にした輸送艦
BA_C：ボディ系のパーツを使用していないデザイン

[サンプルモーション]

SAMP 1：戦闘機が1機のみ。戦闘機の横から見た形がわかりやすい。テスト画像としては最適
SAMP 2：戦闘機1機。きりもみしながら急上昇する。前から見たところ，うしろから見たところがわかりやすい
SAMP 3：戦艦1艘。戦艦に突っ込んでいく戦闘機の視点。戦艦のテスト画像としては最適
SAMP 4：戦艦1艘。かなり戦艦に接近するため，デザインによっては，戦艦の内部に入ってしまうかも
SAMP 5：戦闘機1機,戦艦1艘。遠くの戦艦から，戦闘機が1機発射され手前に飛んでくる
SAMP 6：戦闘機1機,戦艦1艘。戦艦に突入する戦闘機。戦艦のぎりぎり下をくぐる
SAMP 7：戦闘機2機,戦艦1艘。戦艦を護衛するように，平行して戦闘機が2機飛んでいく
SAMP 8：戦闘機1機,戦艦4艘。進行する宇宙艦隊。戦闘機が視点のすぐ側を通過していく
SAMP 9：戦艦1艘。レーザー攻撃を受け,爆発する戦艦。ここで使われているレーザーや爆発は，変更できない

[注意点のまとめ]

・HDから起動しておく
・ディレクトリが入力できないときは，マウスカーソルの位置に注意
・ディレクトリ名は，フルパスで指定する
・command.xにパスを通す

3）FD版システムの作成

HDのない方は，仕方がないのでFD版のシステムディスクを制作しましょう。付録ディスクをドライブ0に入れ，起動します。すると，写真1のメニューが出ますので，

「システムフロッピーを作成する」

をクリックします。すると，

「ドライブ1に空きフロッピーを入れてください
（フォーマットするので内容は失われます）」

と表示されますので，指示に従ってください。「実行」しますと，あとは自動的に展開します。かなり待たされたのち，

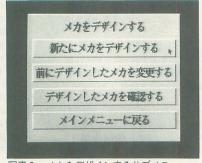
「GENIE起動ディスクの作成が終了しました。
GENIEを実行するときには，このフロッピーから立ちあげてください。」

と表示されたら完成です。

「確認」をクリックすると，再びインストーラのメニュー画面に戻りますので，次にサンプルデータフロッピーを作成しましょう。

FD版の「GENIE」では，ドライブ0にGENIEシステムディスクを入れ，ドライブ1にワークディスクを入れて使用します。ワークディスクは，フォーマット済みで空き容量が十分にあれば，どんなディスクでもかまいません。ですから，このサンプルデータディスクは絶対にないといけないというものではありません。

また逆に，サンプルデータディスクは，1枚作ればそれで十分というものでもありません。サンプルデータのアニメーションをすべて作画・アニメーションさせようとすると，とても1枚のディスクには収まりません。また，あとで詳しく解説するような，形状の差し替えなどを行えば，たくさんのサンプルデータディスクが必要となります。

つまり，完全にオリジナルデータの制作を行う場合はまったく不要で，積極的にサンプルを流用する場合はたくさん用意します。必要に応じて，作成してください。

メニューから，

「サンプルデータフロッピーを作成する」

を選択すると，システムディスクと同様に，

「ドライブ1に空きフロッピーを入れてください」

となります。ディスクを入れて，実行すれば，あとは自動的に展開されます。終了すると，再びメニューに戻りますので，さらにもう1枚作成するなり，終了するなりしてください。

システムディスクとサンプルディスクをドライブ0，1に入れてリセットすれば，「GENIE」が起動します。

[注意点のまとめ]

・サンプルデータディスクがなくても「GENIE」は使える

[3] メカデザインに挑戦

1）メニュー

自由にオリジナルメカをデザインする前に，操作に慣れるために，ひと通り指示に従って，簡単な戦闘機を1つ作ってください。とりあえず，「GENIE」を起動させて，メインメニューを出します（写真2）。

なんとも，シンプルなメニューですね。こんなメニューを見ると，とりあえず害のなさそうな「操作解説」などクリックしたくなりませんか？　クリックしてもかまいませんが，別にたいしたことは書いてありませんから，さっさとウインドウの左上の「/」をクリックして，メインメニューに戻ってください。

それでは本題に戻って，「メカをデザインする」をクリックしましょう。サブメニューが表示されます（写真3）。特に解説するほどのものではありません。

「新たにメカをデザインする」

を選択しましょう。

2）FFEの起動

CGAシステムを使ったことのある人は，出てきた画面に見覚えがありますね。そう，これはFFE.Xです。

ただし，このFFEは，以前配布したものよりバージョンアップされているだけでなく，「GENIE」専用にいくつか改造してあります。まず，「GENIE」では使用しないメニューを青色で表示し，選択できないようになっています。また，視点の位置や，グリッドなどが，「GENIE」の都合のよい値に設定されています（写真4）。

ということで，FFEを使いこなせる方は，ほとんどこの解説を読む必要はありません。斜め読みして，自分の知らなかった機能などを見つけたら，そこだけちゃんと読んでください。

今回は，基礎編ということで，初めての方にもわかるようにひととおり解説していきます。

写真2　メインメニュー

写真3　メカをデザインするサブメニュー

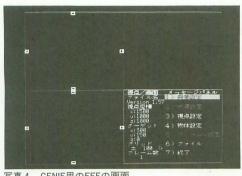



写真４　GENIE用のFFEの画面

図１　FFEの画面構成

FFEの画面は，図１のような構成になっています。

操作の大部分は，メッセージパネルで行います。現在，メッセージパネルには，

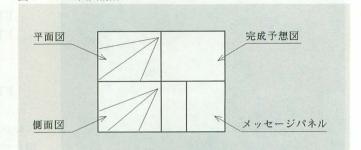
３)視点設定
４)物体設定
６)ファイル
７)終了

という４つのメニューしかありません。青色で表示されている「１)背景設定」などが，先ほど解説した，「GENIE」でいまは使わないメニューです。

３)　パーツを置く：ボディ

「４)物体設定」をクリックします。キーボードで「４」を入力したり，上下カーソルキーで選択してリターンキーを押してもかまいません。

メッセージパネルの表示が切り替わります(写真５)。

変更や削除をしようにも，まだ何も置いていませんから，当然，「１)追加」を選択します。

メッセージパネルは，ファイル選択モードになり，パーツの分類が表示されます(写真６)。

これが，「GENIE」の誇る(？)データベースです。「<>」はディレクトリを意味し，「戦闘機」や「タンク」というディレクトリの下に，それぞれたくさんのパーツがあるわけです。どんなメカがあるかは，14～17ページの一覧表をご覧ください。どうです，これだけあれば，どれを使おうか迷うでしょう(作るほうは大変やってんでぇ)。

とりあえず今回は練習なので，順当なところで<戦闘機>のなかからボディを選びましょう。「<戦闘機>」を

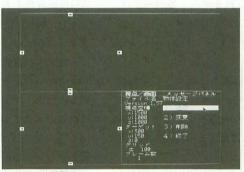



写真５　物体設定

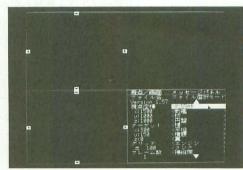



写真６　ファイル選択モード

メカデザインのコツ

・紙と筆記用具

「GENIE」を使う前に，まず紙と筆記用具を用意してください。FFE.Xでは，現在置こうとしているパーツの座標などは表示されますが，すでに配置したパーツの座標などを参照する機能がありません。そこでパーツを，左右対象の位置や，等間隔に並べて置こうとするときに，位置をメモっておくと，大変重宝します。

・左右対象形

左右対象の形状を作るために，左右に同じものを置く場合，その位置は，Ｙ座標だけがプラスとマイナスが逆になります。あたりまえですね。パーツを左右反転しなければならないときは，拡大のＹの値をマイナスにします。これも，あたりまえですね。ちょっと，ややこしいのが回転ですが，Ｙの値は同じで，Ｘ，Ｚ軸回転の値のプラスとマイナスが逆になります。

・回転の注意

通常，Ｘ軸方向に拡大すると，前後に長い物体になります。しかし，Ｚ軸に90度回転してからＸ軸方向に拡大すると，横に伸びます。90度回転したことを忘れて，バグだと騒がないように。また，Ｘ，Ｙ，Ｚ軸の複数を回転させると，結果としてあらぬ方向を向くことがあります。これもバグではありませんので，落ちついて考えましょう。

・パーツの分類にとらわれない

パーツは，〈戦闘機〉や〈エンジン〉などに分類されていますが，この分類にはとらわれないでください。〈戦艦〉のボディを戦闘機のボディにするどころか，〈楯〉や〈タンク〉をボディにしても，なんら問題ありません。サンプルの戦艦「BA_C」なんか，艦橋を２つくっつけてボディにして，砲台を横に引き延ばしてエンジンにしています。

・大きく変形させる

右の写真(SCAL.PIC)をご覧ください。この３つのパーツは，どれも同じパーツを特定の軸方向に拡大，縮小したものです。ご覧のように，まったくイメージが変わります。200足らずのパーツも，大きく変形させたり，さらにそれらを組み合わせることで，何倍もの数のデータにすることができます。

・交差

単にくっつけるだけでなく，たとえば前後に長い(あるいは長くした)パーツと，横に長いパーツを交差させると，また別の形が見えてきます。CGのメリットですから，大胆にめり込ませてみましょう。

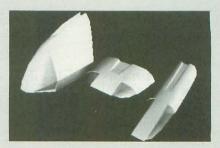
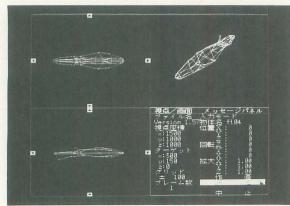



写真7　完成予想図が表示される

クリックします。HD版の場合は,

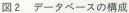
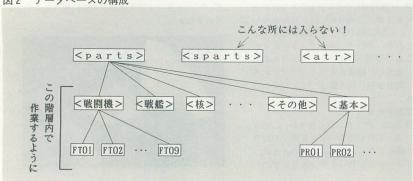
▲
FT01.SUF
FT02.SUF
⋮
⋮
FT09.SUF
▼

のように，9種類のパーツが表示されます。残念ながら，FD版のほうは5種類しか表示されません。各パーツは，面数の少ない順に並んでいますので，メモリやCPUパワーが十分でない方は，できるだけ上のほうのパーツを選択するのが無難です。

ここでは，「FT04.SUF」を選択しましょう。平面図と，側面図に黄色いワイヤーフレームで，FT04が表示されます。もし誤ってほかのパーツを選択してしまった場合には，「中止」をクリックすれば，もとの画面に戻ることができます。

メッセージウィンドウの「作画」をクリックすると，完成予想図のところに斜めから見た図が表示されます。まずは，FT04をこのまま置くことにしましょう。「決定」をクリックします(写真7)。

4)　パーツを置く：ウイング

これだけではちょっと情けないので，ハネの1枚でもつけましょうか。先ほどと同様に，「1)追加」を選択します。すると，

▲
FT01.SUF
FT02.SUF
⋮
⋮
FT09.SUF
▼

おやこれは，さっき戦闘機のボディを選択した状態と同じですね。これは，データベース構成が図2のようになっているからです。上の階層に戻るときは，「<<戻る>>」をクリックします。

さて，ハネのパーツはやっぱり<翼>から選んでみましょうか。「<翼>」をクリックして，今回はちょっと面数の多そうな「WG12.SUF」を使います。

とはいっても，HDでお使いの方は，メッセージパネルにWG01〜WG11までしか表示されていません。そうです，このパネルはデータが多いときはスクロールするわけです。下カーソルキーか，「▼」をクリックすれば，WG12，WG13，それに「<<戻る>>」や「中止」のメニューも出てきます。

「WG12」をクリックすると，再び平面図と側面図にWG12が黄色く表示されます(写真8)。「作画」で確認してもいいのですが，ハネの位置はもう少し後ろのほうがバランスがよさそうです。移動させてみましょう。

WG12の現在位置は，(X，Y，Z)＝(0，0，0)，つまり原点です。位置を移動させるには，マウスで位置を指定する方法と，メッセージウィンドウの「位置X，Y，Z」に数値を入力する方法があります。

まず，マウスで移動させる場合，平面図，側面図の移動させたい位置をクリックします。各図面の座標軸は図3のとおりです。青いグリッドの1マスが100ということを念頭において，(−650，0，0)のあたりをクリックしてください(写真9)。黄色いWG12が消え，クリックした位置に白い小さな「□」のマークが現れます。

本当に(−650，0，0)になったかどうかは，メッセージパネルの座標値を見ればわかります。違っていたら，(−650，0，0)になるまで，平面図か側面図をクリックしてください。

図2　データベースの構成

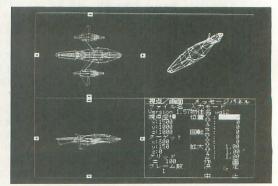



写真8　平面図と側面図にWG12が黄色く表示される

(−650, 0, 0)になりましたら，右クリックします(写真10)。すると，WG12が，移動した位置に再び黄色で表示されます。このように実際の作業では，左クリックで位置を指定して，右クリックで表示してみるというのを交互に行って，適当な位置を探していきます。

本当にこの位置でよいかを見るために，「作画」をクリックしてみましょう。ほら，ずっと戦闘機らしくなりました。最後に「決定」をクリックして確定します(写真11)。

次に，キーボードで位置を入力する方法ですが，この場合，上下カーソルキーでメッセージウィンドウの変更したい座標へ移動したのち，数値を入力します。今回は，X座標を変更するだけですから，「位置X：0」が反転表示している状態で，キーボードから「−650」と入力，リターンキーを押すと，WG12の表示位置が移動します。「作画」「決定」も，上下カーソルで選択してリターンすれば，マウスの場合と同様に使用できます。

つまり，はっきりと数値がわかっているときはキーボード入力，いろいろ試行錯誤するときはマウス入力のほうが楽でしょう。うまく使い分けてください。

5) パーツを置く3：砲台

さて，ボディとウイングだけでもそれなりに見えますが，もう少し凝ってボディの下に砲台を1つ追加してみましょう。

要領はハネのときと同じです。「追加」して，「<<戻る>>」で<翼>から抜け，<砲台>に入ります。「<砲台>」はスクロールしないと画面に現れないので注意してください。「CN04」を選択すると，キャノン砲みたいな形が現れます。

しかし，ボディの下につけたいので，まず向きを上下ひっくり返さないといけません。ひっくり返すには，X軸回りに180度回転する方法と，Z軸方向に−1倍する方法があります。なぜ，−1倍すると反転するかといえば……まぁ，世の中そういうもんなんです。

どちらの方法も手間は変わりません。回転させるなら

図3　各図面の座標軸

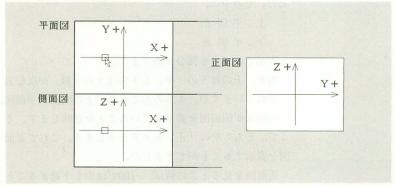
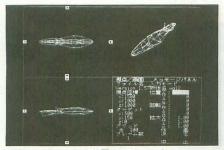



写真9　移動したい位置をクリックする

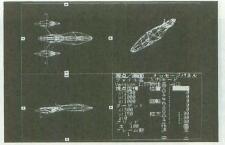



写真10　右クリックすると表示される

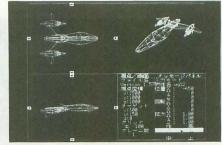



写真11　「作画」をして確認する

注意事項

付録ディスクの展開・実行は，本文をよく読んで行ってください。適当にやってたら，変なことになるかも。知らんでぇ。

・機種による制限

［メモリが1Mバイトしかない場合］

残念ながら起動しません。

［メモリが2Mバイトの場合］

あまり複雑なメカは作れません。比較的簡単な形状でも，それが数多く出てきたりすると作画できません。また，あまり長いアニメーション再生もできません。条件によっては，サンプルデータの一部も作画できないでしょう。

［CPUパワーが低い機種の場合］

一部の機種では，画像の内容によって，毎秒20フレームの滑らかなアニメーションができず，若干の遅れが出ます。

・誤差拡散について

作画設定で「誤差拡散」を「あり」に設定すると，疑似的に6万5千色以上の高画質でレンダリングします。しかし，画像データがきわめて大きくなりますので，状況によって，アニメーションできる時間が短くなる，アニメーション速度が遅くなる，またFD版ではディスク1枚に収まらないといった現象が表れます。

・バグについて

バグなどないように，それなりに注意して開発していますが，それなりにしか注意していませんので，それなりに現れるものと思われます。あらかじめご了承ください。

・無補償

このシステムによって，あなたがいかなる損害を受けようとも，DōGAおよびOh!X編集部は，なんら責任を負わないと思います。あらかじめご了承ください。

・使用制限

このシステムは，多くの方にCGAの楽しさを知っていただくことを目的に，DōGAが採算を無視して制作，配布いたしました。

よって，このシステムやそのデータを，営利活動等に利用することを禁じます(っていうか，そりゃセコイってもんだよ)。

・事前承諾

このシステムやそのデータを使って，CGA作品を制作し，当方が主催するCGAコンテスト，その他のCG・映像系のコンテストに出品することは，なんら問題がありません。その場合，当方に承諾を得る必要もまったくありません。積極的に行ってください。

しかし，以下の行為を行う場合には，事前にDōGAまで連絡をとり，承諾を得てください。

・データなどの一部または全部を，ほかのプログラムなどに流用する

・ほかの機種に移植する

・改変して，配布する

「回転」のXの値のところ，反転させるなら「拡大」のZの値のところを，上下カーソルキーかマウスで反転させて，キーボードから，回転なら180，拡大なら－1を入力してリターンです。残念ながら，回転や拡大は，位置の指定のときのようにマウスだけでは指定できないわけです。

位置もこのままではおかしいので，移動させます。(－450，0，－100)ぐらいでいかがでしょう(写真12)。

すると，ここでちょっと問題が発生します。砲台がボディにちゃんとつながっているか，ハネがじゃまでよく見えません。そこで，正面図を出してみましょう。

メッセージパネルの左上にある「視点/画面」をクリックします。すると左側の下の部分の表示内容が変わりました。その中に，

上：平 側 正
下：平 側 正

と表示されている部分があります。

現在，上のほうの「平」と下のほうの「側」が反転表示されていますが，もちろんこれは，上の図面が平面図，下の図面が側面図を表示していることを意味します。そこで，どちらかの「正」をクリックします。これで正面図を表示することができました。

正面図を見るとZの値は，－100では少し下過ぎることがわかります。－50に変更しましょう。これで「決定」です(写真13)。

ところで，ついでに解説すると，この図面切り替え表示のすぐ上に，

カウント
▲　50▼

という表示があります。このカウントというのは，マウスで指定する位置の精度です。現在のように50のままだと，100とか，250といった値になり，110，248といった値にすることができません。そんな必要があるときは，「▲」をクリックしてカウントの値を小さくします。しかし，あまり小さくしすぎると，ぴったり1000，あるいはずばり0にしようとしたとき，逆に面倒くさくなります。

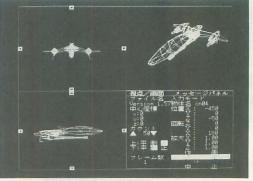
6) パーツの変更

以上，3つのパーツを組み合わせてみました。もっとたくさんくっつける場合も要領は同じです。同じことをしてても仕方がないので，今度は，一度設定したパーツを変更させるという作業を行ってみます。「1)追加」ではなく，「2)変更」に入ってください。

「選択モード」と表示されると思います。これは，どのパーツについて変更するのか，まずパーツを選択しなければいけないということです。パーツの選択は，平面図や側面図で，マウスで変更したいパーツの中心付近をクリックしてやります。今回は，先ほど設定したハネ(WG12)を選択します。

写真12　X軸回りに180度回転させ，位置も変更

写真13　正面図を見ながら位置を確認する

しかしながら，複数のパーツが近くにある場合，本当にハネが選択されたかどうか疑問です。そのために，メッセージパネルの中ほどの「物体名」横に，現在選択された物体名が表示されるようになっています。もし，ちゃんと「WG12」と表示されずに，「FT04」とか「CN04」になっていれば，正しく選択されて

付録の付録のMODEL.X

「GENIE」は，初心者向けのシステムということで，パワーユーザーの皆さんにはちょっともの足りなかったかもしれません。そこで，そんなあなたのために，付録ディスクの付録として，新しいモデリングツール「MODEL.X」をプレデビューさせます。

このモデリングツールは，Ko-WINDOWの作者として有名な小林君が，CGAシステムver.3への布石として開発したものです。このMODEL.Xは，従来のモデリングツールCAD.Xとは，いろんな面で異なります。

たとえば，CAD.Xはそのソースが紛失したこともあり，内部公開どころか，バージョンアップもほとんど行われないという完全なブラックボックスとなっていました。それに対してこのMODEL.Xは，ごく基本的なベースと，マクロ言語，サンプルプログラムが用意されているだけで，各機能は，そのマクロ言語を記述することで，ユーザーの皆さんがよってたかって開発するということが前提になっています。つまり，成長型のシステムというわけです。

マクロ言語で記述された各機能は，MODEL.X起動時に組み込まれますので，新たに機能を追加したり，各機能の一部を変更したりしても，MODEL.X全体を再コンパイルする必要がありません。つまり，インタプリタといえます。

今回はプレデビューということで，ほとんど機能はついていません。というか，近く正式にデビューさせるために，皆さんにこのマクロ言語でいろんな機能を開発して送ってほしいというのが本音です。興味がある方は，付録ディスクの「MODEL」というディレクトリの中のファイルを展開して，ドキュメントなどを読んでみてください。そして，開発に参加してくださる方は，当チームの「MODEL開発係」までご連絡ください。

なお，現段階のMODEL.Xもちゃんと起動しますし，領域指定による複数のポリゴンの移動，削除など，CAD.Xにはなかった機能もありますので，プログラムができない方も使ってみてください。そのうえで，要望・提案書などを送ってくだされば幸いです。

DōGA

いないということです(写真14)。

その場合，もう一度平面図，側面図で指定し直すか，「物体名」のすぐ下の「選択」をクリックしてください。「物体名」が，指定した位置に近いほかのパーツに切り替わります。さらに「選択」をクリックすると，また次に近いパーツに変わりますので，パーツ数が少ない場合は，結構簡単に探すことができます。

ちゃんとWG12が選択できれば，「決定」します。平面図，側面図などのWG12が黄色く表示され，「2)追加」したときと同じ状態に戻ります。

ここでは,ハネを少し横に大きくします。「拡大」の「Y」のところに「1.3」という値を入力しましょう。そして，「作画」で確認し，問題がなければ「決定」します。

7) セーブ・終了

新しいメカができれば，さっそくセーブしましょう。「4)終了」で物体設定のモードを抜けて,「6)ファイル」に入ります。

「SAVE」のメニューしかないので，選択するのはもちろん「SAVE」で，「フレームソース(FSC)」のメニューしかないから，もちろん「フレームソース(FSC)」を選びます。このあたりの操作には無駄が多いのですが，FFE.Xを「GENIE」に流用したのが原因ですので，あきらめてください。

「名前の入力」と表示されたら，適当にキーボードから入力してください。今回は「OHX01」とでもしておきましょう。名前のつけ方は，必ず8文字以内，できれば5文字以内にしてください。

セーブが終わると，メインメニューに戻りますから，「7)終了」でFFEを終了します。すると，しばらくごそごそ作業をしたあと,「GENIE」のメニュー画面に戻ります。

以上で，新しい戦闘機が1機デザインできたわけです。いかがでしたか？　文章で解説すると非常に長いのですが，実際に作業を行ってみるとたった1分30秒(実測)です。1分30秒で，戦闘機1機デザインできる！　そんなシステム，いままでにありましたか？

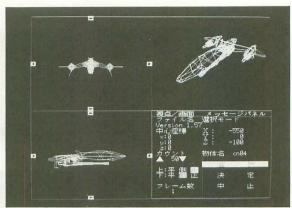



写真14　選択された物体名が表示される

[4] メカの確認

メカデザインが終了したら，ワイヤーフレームではなくて，ちゃんとレンダリングした絵を見てみたいでしょう。まかせてください，そんな機能もつけておきました。

現在「GENIE」の画面は，メカをデザインするサブメニュー(写真3)になっていると思います。このなかの下から2番目の「デザインしたメカを確認する」をクリックしてください。

「確認するメカを選択してください」と表示され，「FI_A」や「BA_A」などのサンプルデータに交じって，先ほどセーブした「OHX01」があるはずです(写真15)。このボタンをクリックします。

あとは全自動。先ほどの戦闘機をいろんな角度から見た画像を，5枚作画してくれます(写真16)。どうです，レンダリングすると，また違ったイメージになったでしょう。そして，作画が全部終わると，それらの画像の連続表示に入ります。

ついでに，連続表示中の機能について紹介しましょう。まず，スペースキ

写真15　メカデータの一覧

付録の付録の付録

ただいまOh!X編集部より，付録ディスクにまだ少し空き容量があってもったいないので，何か入れてくれという連絡がありました。そこで，CGAシステムの画像データを，Oh!X1994年3月号の付録ディスクのAMI.X用に変換するツール「TCH2AMI.X」を発表します。

AMI.Xは，福嶋章太氏が制作された，SCSI装置のアニメーション再生プログラムで，MOに収められた大量の画像を，きわめて高速に，長時間再生することができるという優れもののツールです。

3月号には，CGAシステムからのコンバート方法が紹介されていました。しかし，複数のカットを連続してアニメーションしようとするとかなり複雑になり，タイムチャートによって記述される止め絵(.wait)や繰り返し(.repeat)などにも対応していませんでした。そこで，もっと手軽に，タイムチャートを指定するだけで一括してコンバートしようということで，このツールを制作しました。

[使用法]

tch2ami [オプション] <タイムチャートファイル> -O<AMIファイル>

オプション:

-C<出力画面サイズ>,<色数>
　出力するAMIファイルの大きさと色数を指定
　出力画面サイズは64,128,256から選択
　色数は256,65536から選択
　デフォルトは64,65536

-I<入力画面サイズ>
　入力画面の大きさを指定
　　64,128,256 から選択
　　デフォルトは入力画像全面

-P<入力画面位置>
　入力画面のうち変換を行う部分の左上座標を指定
　　デフォルトは 0,0

-A
　縮小時にアンチエイリアスをかける

タイムチャートファイルの拡張子.tchは省略可能です。

AMIファイルの拡張子.amiは省略可能です。

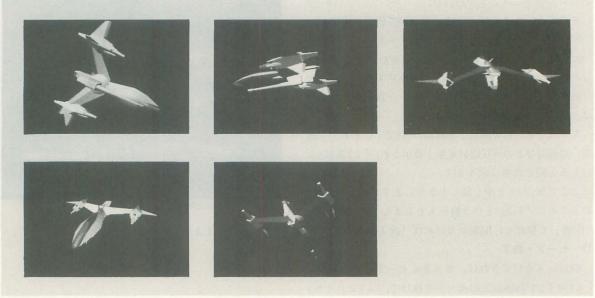



写真16　自動で作成された5枚の画像

ーが，一時停止，コマ送りとなっています。そして，一時停止の状態から抜けるのがリターンキーです。現在連続表示は，1枚あたり0.5秒に設定されていますが，ROLL UP，ROLL DOWNキーで変更することができます。また，UNDOキーでもとの設定に戻ります。

そのほか，HOMEキーは，1枚目の画像を表示して一時停止，アルファベットの「L」は31kHzと15kHzを切り替えます。ビデオボードなどで録画する場合に役に立つでしょう。

そして最後に，見飽きたらESCキーで終了します。

写真17　アニメーション作成のサブメニュー

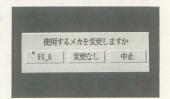



写真18　形状差し替え機能

ユーザーの皆さんへ

この「GENIE」は，たいした技術は使われておらず，アイデアだけのシステムといえます。こういったDōGAらしいアイデアは，なんといってもユーザーとの交流から生まれてきます。DōGAでは，これからも皆さんからのご意見，ご感想などをお待ちしております。

1）　GENIEコンテスト

皆さんが「GENIE」で作ったデザインを送ってください。こんなかっこいいメカができた，意外なパーツを意外な使い方をしてこんな形を作った，宇宙戦闘機，戦艦以外のこんなものを作ったなど，面白い作品を9月号で紹介したいと思います(賞品は……あてにしないでください)。締め切りは，7月10日必着。よろしく。

2）「GENIE」追加データ集について

今回のパーツの数は十分だったでしょうか？　いろいろと足りない形があるのは承知していますが，パーツの数の少なさに不満を感じる前に「GENIE」自体に飽きてしまうだろうということで，現時点では追加データ集を出す計画はありません。しかし，追加データ集の要望が多いようでしたら検討します。

希望される方は，現在よく使用するパーツ名や，どんな形状が欲しいのかを(できればラフでもいいから紙に描いて)送ってください。また，CAD.Xで作ったパーツも募集します。

「GENIE」のメニューに戻りますので，「メインメニューに戻る」でメインメニューに戻ってください。

[5] 差し替えアニメーション

せっかくオリジナルメカをデザインしたのですから，「メカを確認する」の5枚の画像だけではもの足りないでしょう。サンプルデータを流用して，アニメーションさせてみましょう。

まず，メインメニューから「アニメーションを作る」を選択します。

写真17のメニューが出ますので，「作画」をクリックしてください。サンプル用のモーションが「SAMP1」～「SAMP9」まで9種類表示されます。どれを選ぶかは，コラムの「サンプルデータの解説」を読んで決めていただければ結構ですが，戦闘機が1機出てくるだけのものは，「SAMP1」と「SAMP2」の2つしかありませんので，今回は「SAMP1」を選んでみましょう(ほかのサンプルでも可能です)。

すると，

「使用するメカを変更しますか」

と聞いてきます(写真18)。これが「GENIE」の形状差し替え機能です。一度設定された動きのデータに対して，メカを差し替えていろんなアニメーションを作れるようにしているのです。

写真18の表示は，「SAMP1」という動きのデータの中には「FI_A」というメカが1種類だけ使われている，ということを意味します。そこで，この「FI_A」を先ほど制作した「OHX01」に差し替えてみましょう。

「FI_A」をクリックすると，「どのメカに変更しますか」と聞いてきます。そして，ほかのサンプル「FI_B」や「BA_A」の最後に「OHX01」が表示されています(写真19)。

これをクリックします。

すると，写真20のように「他にも変更がありますか」と聞かれます。ほかにもありますかといわれても，ほかにメカはありません。「変更なし」をクリックします。

すると，次に写真21のメニューが出ます。色調の変更とは何でしょうね。

先ほど「メカを確認する」で作画させたときは，白っぽい色になっていたと思います。各パーツは，どれも白を基調にして制作されているからです。でも，みんな白っぽいのは寂しいとか，戦闘シーンで，敵，味方の区別をつけるために色調を分けたいとか，戦艦は暗い色，戦闘機は白のままにしたいといった要望もあるでしょう。そんなとき，そのメカの色調を変更するわけです。

「OHX01　白」をクリックすると，色の選択肢が「白」「青」「赤」「緑」「紫」と5色表示されます。好きなのを選んでください。「他にも変更がありますか」と聞いてきますが，もちろん「変更なし」です。

この色調変更ですが，たとえば「OHX01」が5機編隊飛行しているというカットを制作した場合，それら5機は必ず同じ色調になります。少佐の乗っている機体だけは赤，などということはできません。

などといっているうちにも，作画はどんどん進んでいると思います。あとは自動的に作画を行いますので，ただ待つしかありません。この「SAMP1」は30フレームありますので，X68000XVIの場合で10分ぐらいかかるでしょう。

作画計算が終わると，「アニメーションを作る」のメニューに戻ります。早速，できたアニメーションを再生しましょう。「アニメーション再生」をクリックします。「アニメーションを選択してください」と表示されますので，当然「SAMP1」を選択してください。しばらく，読み込みを行ったのち，短いながらもなめらかなアニメーションが表示されます。いかがですか。こういった感じでオリジナルメカが飛び回れば，楽しいでしょう。

なお，このアニメーション再生中の機能は，「デザインしたメカを確認する」の連続表示と同じことが行えます。気に入ったら，ビデオにでも収録してください。

[6] おわりに

2年前の付録ディスクとして，CGAシステムを発表したとき，「お試しシステム」というものがついていたのを覚えているでしょうか？

「ワゴン車」，「帆船」といった形状のメニューと，「まっすぐ走ってくる」といった動きのメニューを選択，実行するだけで，アニメーションができるというものでした。文字どおり，誰にでも自分のディスプレイ上でCGAを体験してもらい，CGAに興味をもってもらうという目的を果たすには十分で，非常に好評でした。

しかしながら，当方にはいくつか不満がありました。まず，あの「お試しシステム」は基本的に，すでにできているデータを再生するだけで，なんら創造性というものがありません。また，CGAシステムとの隔たりが大きくて，「お試しシステム」が使えるようになったからといって，CGAシステムが使えるようには全然ならないという問題です。現に，すぐ飽きてしまったり，CGAシステムを使うには至らなかった人が大勢いました。

今回発表した「GENIE」は，それらの問題を解決するために開発された「お試しシステム」の第2弾ともいえます。LESSON 1：お試しシステム，LESSON 2：GENIE，LESSON 3……と続けていくうちに，何の壁にもぶち当たらないで，CGAシステムという高台につながる階段を登っていく。そういったCGA入門者のための環境を提供したいわけです。では，LESSON 3は？　それはまた，そのうちに……。

来月号は，まだ解説をしていない「動きをデザインする」を中心に，「GENIE」の応用編を行いたいと思います。皆さんも高台めざしてがんばりましょう。

＊　＊　＊

ところで，最近のテレビCMのなかで，第6回アマチュアCGAコンテストのビデオの一部を使用しているものがあります(もちろん当方の許可済み)。時間にして0.数秒。気がついた方は，まずいらっしゃらないでしょう。ヒントは，「マルチメディア」です。暇な方は探してみてください。

では，来月。

写真19　メカデータの一覧から選択

写真20

写真21　色調の変更のメニュー

問い合わせ先

CGA入門キット「GENIE」は，プロジェクトチームDōGAが企画，制作し，Oh!Xを通じて配布しております。プログラムおよびデータの著作権は，すべてプロジェクトチームDōGAが管理・所有しています。したがって，内容に関するお問い合わせは，プロジェクトチームDōGAまでお願いします。

〒533　大阪市東淀川区淡路5-17-2　102号
プロジェクトチームDōGA「GENIE」係

ただし，メディアに関するお問い合わせ，たとえばディスクがクラッシュしていたというトラブルなどは，Oh!X編集部へお願いします。

また，3.5インチへのコンバートを希望する方は，
・送り先を明記し，切手を貼った返信用封筒
・フォーマット済みの3.5インチ2HDディスク
を同封して，

〒103　東京都中央区日本橋浜町3-42-3
ソフトバンク株式会社
Oh!X編集部「GENIE」コンバート係
までご連絡ください。

"実戦！"ゲーム作りの

KNOW HOW

〈基本セオリー編その1〉

スティック判定とDDA

Taguchi Atsushi 田口 敦

予告とはちょっと違ってしまいますが，今回は小物テクニック集その1ということでジョイスティックの判定法とデジタル微分解析による方向探索を扱ってみます。

当初，隔月を基本とした不定期連載ということで連載を開始したのですが，連載3回めにしてさっそく不定期になってしまいましたねぇ。私事の都合とはいえ，期待してくれた読者の方には非常に申しわけないと思っています。しかぁし，私は「ごぅいんぐまいうぇい野郎」なので当分上記のペースで連載を続けていくと思います。と，あんまり偉そうなこともいってられないので，一応対策は練ってあります。どのような対策かというと「マルチプロセッサ化」です。ということで(?)，原稿があがりそうにないときは"助っ人君"に頼むことにしました。

これで少しは記事の安定供給ができると思います（できるといいなぁ）。

とまあ，読者にとってはとてもつまらない書き出しになってしまいましたね。気分を変えて，ちょっとだけ新コンシューマハードとゲームについての脱線話をひとつ。

私こと田口は，4月をもって学生から社会人に変態しました。"ゲーム作りのKNOW HOW"なんて記事を書いているくらいですから，当然就職先はゲームソフトの制作会社です。

いままで新ハード群について，雑誌の情報をもとにわけ知り顔で，3DOがどうしただの，PSXがこうしただのと勝手な評価をしていたわけですが，実際にそれらの仕様書を見てみるとなかなかどうして，凄いじゃないですか。新ハード間の性能の差というのははっきりいって問題にしないほうがよいかもしれません。どれをとっても従来のハードから飛躍しているのですから（ハード自体いままでと概念が異なる部分が多い）。パナソニックのCFでコンピュータの天才というのが出てきますが，伊達じゃないって感じですね。

3DOではすでにゲームらしきものが多数発売されていますが，ほとんどの場合従来の評価の仕方では手に余る作品ばかりですね。それがよいことなのか，悪いことなのかは時間が証明してくれるとして，とりあえず，ソフト面でも変化が起きていることは確かです。あえて，マルチメディア云々などというつもりはありませんが，その方向性はユーザーと制作者の意思により，大きく変わると思います。

早い話が，この記事を読んでいるあなた方次第でどうにでもなる世の中になってきているということです(誇張度60%)。ということで，ルールの確立していない現在，自分の道を突っ走るというのもいいんじゃないかなと思っています。

おっと，ちょっとわけわからんことをいって脱線しすぎたかな。道を進むときにこの記事が道標になったら最高っすね。

◆今回やること

前回，BGのマッピング簡略編ということで簡単なスクロール処理をやってみたのですが，前回の予告によると今回応用編ということになっていました。当初，私もその予定でオリジナルを含めたサンプルを制作していたのですが，"オリジナル"って部分で結構手間を食ってしまい，原稿化にはもう少しかかりそうなので今回は少し基本に戻ってセオリー編としたいと思います。

セオリー編といっても甘く見てはいけません。基礎ってものがいちばん難しいのです。ほとんどの初心者がそこを超えられずに断念してしまうのです。

また，ある程度の中級者（あくまでもアマチュアレベルでの）でも，セオリーを知らないために，プログラムをより困難にしている場合があります。

しかし，今回のサンプルはセオリーのなかのわずかな部分であり，あまり威張れるものではないのですが，ゲームプログラムを行ううえで少しは参考になるのではないかと思います。プログラムのコード自体は相変わらず激甘ですが，そこは著者のいたらぬとこと笑ってやってください（といっても笑わないのが"お・と・な"ってもんである。うん）。

ということで，今回は実際にゲームによく使用され，しかもいろいろ応用のできる「ジョイスティックのデータ判定」と「方向検索」の2本立てでいきたいと思います。

◆ジョイスティックのデータ判定

さっそく本題に入りましょう。

はたしてジョイスティックのデータ取りにセオリーなどあるのかといわれるかと思いますが，一応それなりのものが存在するのです。

リスト1を見てください。アマチュアの同人ゲームなどでよく見かけるプログラムです。このプログラムは1つひとつジョイスティックデータを調べていくという一見無駄のないプログラムに見えますが，このプログラムだと，上を押された場合と左上が押された場合の負荷が違うのがわかると思います(図1)。特にリアルタイムゲームでは確率的な平均負荷より，最大負荷の重さが評価されるのでリスト1のようなプログラムはゲームにはあまり向いていません（C言語でswitch文やif文を使って記述するとこのようになる）。

で，どうしたらよいかというと，結論としてはリスト2のようなものが最良と思われます。リスト2はジョイスティックの操作に応じてスプライトなどのキャラクター移動値をd2にX値，d3にY値を拾ってくるものです。パッと見で理解できるでしょうか。

これはジョイスティックのデータをインデックス情報としてリスト後半のデータテ

ーブルから移動値を引っ張ってきているのです。このプログラムで注意してなければならないのは，ジョイスティックは押されたときに0を返すという性質を考慮してデータを反転しているということです（反転しなくても判定はできるが)。

この反転をさせるときに数値を反転させるつもりでNEG命令を使用してはならないということです。あくまでもビット単位での反転です。NEGを使用すると2の補数をとってしまうので全然違う値になってしまうのです。

そして，リスト3（図2）です。

リスト2の17行目からを変更します。

このリストはリスト2の方法を使用してリスト1と同じ処理を行ったものです。

どうですか。ずいぶんすっきりした形になったと思います。

これは，リスト2で移動データのおいてあったところに任意のアドレスを配置しておいてJMP命令で引っ張ってきたアドレスに飛ぶって感じになっています。

ここまでくるとかっこいいですね。アセンブラレベルでのプログラミングに醍醐味を感じます。ちなみに，C言語でもポインタの配列をうまく使えば記述できます。

各自このプログラムを参考に自分のスタイルにアレンジしてください。

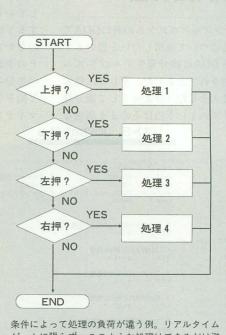
◆ 方向検索

さて，次のサンプルはシューティングゲームには欠かせない方向検索です。

一応説明をしておきますが，ここでいう方向検索というのは座標1に対して座標2がどの方向にあるかということを調べて，

リスト1 joyget_1.s

```
==================  joyget_1.s  ==================
     1: *
     2: *          ジョイスティックデータの判定
     3: *                          (アマチュアのプログラムに良く見られる例)
     4: *
     5: *          (注) このプログラムを実行しても何も起こりません
     6: *
     7:             .include        iocscall.mac
     8:             .include        doscall.mac
     9:
    10: joy_data_get:
    11:             move.w  #0,d1
    12:             IOCS    _JOYGET
    13:
    14:             andi.b  #$0f,d0
    15:                             *RLDU
    16:             cmpi.b  #%1110,d0           * 上が押された場合
    17:             beq     joy_up
    18:             cmpi.b  #%0110,d0           * 右上が押された場合
    19:             beq     joy_ru
    20:             cmpi.b  #%0111,d0           * 右が押された場合
    21:             beq     joy_rg
    22:             cmpi.b  #%0101,d0           * 右下が押された場合
    23:             beq     joy_rd
    24:             cmpi.b  #%1101,d0           * 下が押された場合
    25:             beq     joy_dw
    26:             cmpi.b  #%1001,d0           * 左下が押された場合
    27:             beq     joy_ld
    28:             cmpi.b  #%1011,d0           * 左が押された場合
    29:             beq     joy_lf
    30:             cmpi.b  #%1010,d0           * 左上が押された場合
    31:             beq     joy_lu
    32:             bra     joy_quit            * 何も押されなかった場合
    33:
    34: joy_up:
    35:             * 上が押された場合の処理
    36:             bra     quit
    37: joy_ru:
    38:             * 右上が押された場合の処理
    39:             bra     quit
    40: joy_rg:
    41:             * 右が押された場合の処理
    42:             bra     quit
    43: joy_rd:
    44:             * 右下が押された場合の処理
    45:             bra     quit
    46: joy_dw:
    47:             * 下が押された場合の処理
    48:             bra     quit
    49: joy_ld:
    50:             * 左下が押された場合の処理
    51:             bra     quit
    52: joy_lf:
    53:             * 左が押された場合の処理
    54:             bra     quit
    55: joy_lu:
    56:             * 左上が押された場合の処理
    57:             bra     quit
    58: joy_quit:
    59:             DOS     _EXIT
```

図1 よく見られる例

条件によって処理の負荷が違う例。リアルタイムゲームに限らず，このような処理はできるだけ避けたほうがよい

リスト2 joyget_2.s

```
==================  joyget_2.s  ==================
     1: *
     2: *          ジョイスティックデータの判定
     3: *                          (データをインデックスに使う)
     4: *
     5: *          (注) このプログラムを実行しても何も起こりません
     6: *
     7:
     8:             .include        iocscall.mac
     9:             .include        doscall.mac
    10:
    11: joy_data_get:
    12:             move.w  #0,d1
    13:             IOCS    _JOYGET
    14:             eori.b  #$ff,d0             * インデックスに使用出来るように反転する
    15:             andi.l  #$0f,d0
    16:
    17:             lea.l   xy_data,a1
    18:             add.w   d0,d0               * インデックス情報を
    19:             add.w   d0,d0               * 4倍する
    20:             move.w  0(d0.w,a1),d2       * X移動値を拾う
    21:             move.w  2(d0.w,a1),d3       * Y移動値を拾う
    22:
    23: quit:
    24:             DOS     _EXIT
    25:
    26:             .data
    27:             .even
    28: xy_data:                *X  Y       RLDU
    29:             .dc.w        0, 0     * 0000 何も押されない
    30:             .dc.w        0,-1     * 0001 上
    31:             .dc.w        0, 1     * 0010 下
    32:             .dc.w        0, 0     * 0011 上下 (実際にはありえない)
    33:             .dc.w       -1, 0     * 0100 左
    34:             .dc.w       -1,-1     * 0101 左上
    35:             .dc.w       -1, 1     * 0110 左下
    36:             .dc.w        0, 0     * 0111 左上下 (実際にはありえない)
    37:             .dc.w        1, 0     * 1000 右
    38:             .dc.w        1, 1     * 1001 右上
    39:             .dc.w        1,-1     * 1010 右下
```

それをXとYの移動量に変換するアルゴリズムです。

敵キャラが自機に向かって弾を出す際にどっちの方向に弾を移動させていいのか迷った経験はありませんか。この方向検索というのはなかなか奥が深くて，突き詰めればそれだけで原稿1本あがってしまう代物です。方向検索は基本的にはグラフィックVRAMに線を引くアルゴリズムに似ています。というより，そのまま適用できることもあります。

これには昔からいろいろな方法が考えられていて，単純な8方向検索や三角関数を使う方法など，そのときの必要に応じて使用されてきました。今回はその中から“デジタル微分解析(DDA)”というアルゴリズムを使用したいと思います。

◆ DDAアルゴリズム

まず，サンプルプログラムの前にDDAアルゴリズムについて説明したいと思います。

もともとDDAは線分発生アルゴリズムとして多用されてきました。そして，偉大な先人は考えました。「ひょっとして線分発生アルゴリズムというのはそのまま高速な方向検索に使えるのではないか？」と。線分発生のアルゴリズムはいくつもありますが，このDDAというのはアルゴリズムが非常に簡単であり，しかも高速であるという利点があります。そこでリアルタイムゲームに採用するには最適なんではないかと判断したわけです。

では，実際のアルゴリズムについて説明しましょう。まず，図3を見てください。

この図は与えられた2点のX座標およびY座標の差分を表したものです。つまり，

X2－X1/Y2－Y1

ということになります。

実際のコンピュータの座標に合わせるために左上を基準にしています。

とりあえず，サンプルとしてX幅10，Y幅3の斜線を引く場合を考えてみましょう。

図3と表1を見比べてください。X座標の動きを見てみると常時1ドットずつ変化しているのがわかると思います。これに対するY座標はときどき変化しています。このなかで注目しなければならないのが変数eの挙動です。eには初期値として，X幅の半分の値が入っています。あとはeからY幅を引いてプラスだったらループを返す，マイナスだったらeにX幅を足してY座標を変化させる，というのを繰り返せばいいのです。あとはループごとにX座標をインクリメントさせればOKです

ただし，注意しなければならないのが，このままのアルゴリズムでは，

X幅≧0

Y幅≧0

X幅≧Y幅

という条件でしか使用できないことです。これはプログラム化のときにうまく変換しなければなりません。

◆ サンプルプログラム

では，実際のサンプルプログラムです。今回は，ゲームにそのまま組み込めるように，用途に応じて2パターン作ってみました。

まずは，リスト4の説明です。

このプログラムは座標2を指定した移動量分だけ座標1の方向に近づけるというものです。

30行目あたりまでが，このプログラムを実際に使用したサンプルです。使い方はこれでわかるでしょう。

さて，32行目からが実際のルーチンなわ

リスト3 joyget_3.s

```
==================  joyget_3.s  ==================
     1:          lea.l    adr_tbl,a1
     2:          add.w    d0,d0              * インデックス情報を
     3:          add.w    d0,d0              * 4倍する
     4:          move.w   0(d0.w,a1),a1      * ジャンプアドレスを拾う
     5:          jmp      (a1)
     6:
     7: joy_not:
     8:          * 何も押されなかった場合の処理
     9:          bra      quit
    10: joy_up:
    11:          * 上が押された場合の処理
    12:          bra      quit
    13: joy_dw:
    14:          * 下が押された場合の処理
    15:          bra      quit
    16: joy_lf:
    17:          * 左が押された場合の処理
    18:          bra      quit
    19: joy_lu:
    20:          * 左上が押された場合の処理
    21:          bra      quit
    22: joy_ld:
    23:          * 左下が押された場合の処理
    24:          bra      quit
    25: joy_rg:
    26:          * 右が押された場合の処理
    27:          bra      quit
    28: joy_ru:
    29:          * 右上が押された場合の処理
    30:          bra      quit
    31: joy_rd:
    32:          * 右下が押された場合の処理
    33:          bra      quit
    34:
    35: quit:
    36:          DOS      _EXIT
    37:
    38:          .data
    39:          .even
    40:
    41: adr_tbl:                    * RLDU
    42:          .dc.l    joy_not   * 0000 何も押されない
    43:          .dc.l    joy_up    * 0001 上
    44:          .dc.l    joy_dw    * 0010 下
    45:          .dc.l    0         * 0011 上下（実際にはありえない）
    46:          .dc.l    joy_lf    * 0100 左
    47:          .dc.l    joy_lu    * 0101 左上
    48:          .dc.l    joy_ld    * 0110 左下
    49:          .dc.l    0         * 0111 左上下（実際にはありえない）
    50:          .dc.l    joy_rg    * 1000 右
    51:          .dc.l    joy_ru    * 1001 右上
    52:          .dc.l    joy_rd    * 1010 右下
```

図2 インデックスジャンプの参考例

▶6月号の96ページと100ページのメッセージの差。ウムム……。 島田 尚司(27)新潟県

けです。その後37行目まで，

X2－X1/Y2－Y1

を行っているのはコメントのとおりです。

そして43～55行までがアルゴリズムの説明のところで注意を促した変換部分です。つまり，

X幅≧0

Y幅≧0

X幅≧Y幅

の判定をしているのです。

特に難しいことはしていませんので，そのままプログラムを読んでください。ちなみに，読みやすくするためにメモリにワークをとっていますが，レジスタなどを使えば高速化ができると思います。

んでもって58行目からDDAの本体です。これは特に説明しません。表1と見比べながらコードの流れを追ってみてください。ソースコードデバッガを持っていたらそれを使用すれば簡単です。普通のデバッガでも当然OKです。

もしかしてデバッガを持っていない人&使用方法がわからない人はいますか。デバッガはアセンブラ開発には必需品です。持っていないのでしたらアセンブラの勉強をする前にあらゆる手段を駆使（あらゆるっていってもやりすぎはいけないぜ）して手に入れておきましょう。

72～83行は変換したデータを再変換して元に戻しているだけです。

そして84行からできたデータを元の数値から引いてリターンっと。

このプログラムをサブルーチンとして独立させたいときは32行目からサクっと切ってxdefかなにかで“DDA_1”を外部宣言すれば，そのままほかのプログラムとリンクして使えます。

次にリスト5の説明を軽くします。

基本的にはリスト4とあまり変わりませ

図3 XY差分のグラフ

DDAを使用した場合（0,0）－（3,10）の線分は
このように書かれる

表1 数値の実際の動き

```
作業用変数としてeを使用します
e＝5　wx＝10　wy＝3
e＝e－wy　e＝2　x＝0　y＝0
e＝e－wy　e＝-1　x＝1　y＝0
eがマイナスになった
e＝e＋wxy＝y＋1　（e＝9　y＝1）
e＝e－wy　e＝6　x＝2　y＝1
e＝e－wy　e＝3　x＝3　y＝1
e＝e－wy　e＝0　x＝4　y＝1
e＝e－wy　e＝-3　x＝5　y＝1
eがマイナスになった
e＝e＋wxy＝y＋1　（e＝7　y＝2）
e＝e－wy　e＝4　x＝6　y＝2
e＝e－wy　e＝1　x＝7　y＝2
e＝e－wy　e＝-2　x＝8　y＝2
eがマイナスになった
e＝e＋wxy＝y＋1　（e＝8　y＝3）
e＝e－wy　e＝5　x＝9　y＝3
e＝e－wy　e＝2　x＝10　y＝3
```

リスト4 DDA_1.s

```
==================  DDA_1.s  ==================
    1: *
    2: *         方向検索ルーチン    デジタル微分解析機アルゴリズム
    3: *                            ( Digital Differential Analyzer)
    4: *
    5: *         座標2を移動スピード分だけ座標1に近付ける
    6: *
    7: *         引数    a1          = 座標1のポインタ
    8: *                      X1.w
    9: *                      Y1.w
   10: *                 a2          = 座標2のポインタ
   11: *                      X2.w
   12: *                      Y2.w
   13: *                 d1.w = 移動スピード
   14: *         返り値  a2          = 移動後の座標2のポインタ
   15: *                      X2.w
   16: *                      Y2.w
   17: *
   18:
   19:           .include        doscall.mac
   20:
   21:           move.w  #10,d1
   22:           lea.l   POINT1,a1
   23:           lea.l   POINT2,a2
   24:           jsr     DDA_1
   25:
   26:           DOS     _EXIT
   27:
   28: *                  X   Y
   29: POINT1    .dc.w   000,000
   30: POINT2    .dc.w   010,003
   31:
   32: DDA_1:
   33:           movem.l d0/d2-d5,-(sp)
   34:           move.w  0(a2),d4            * X座標をレジスタに読み込む
   35:           sub.w   0(a1),d4            * X2－X1
   36:           move.w  2(a2),d5            * Y座標をレジスタに読み込む
   37:           sub.w   2(a1),d5            * Y2－Y1
   38:
   39:           move.w  #0,X_neg            * 判定ワークの初期化
   40:           move.w  #0,Y_neg
   41:           move.w  #0,Y_X
   42:
   43: XN_C:     tst.w   d4
   44:           bpl     YN_C
   45:           neg.w   d4
   46:           move.w  #1,X_neg
   47: YN_C:     tst.w   d5
   48:           bpl     XY_C
   49:           neg.w   d5
   50:           move.w  #1,Y_neg
   51: XY_C:
   52:           cmp.w   d4,d5
   53:           bcs     DDA_start
   54:           move.w  #1,Y_X
   55:           exg.l   d4,d5
   56:
   57:           * ここがDDAの本体
   58: DDA_start:
   59:           clr.w   d2
   60:           move.w  d1,d0
   61:           subq.w  #1,d0
   62:           move.w  d4,d3
   63:           lsr.w   #1,d3
   64: DDA_loop:
   65:           sub.w   d5,d3
   66:           bpl     DDAS_00
   67:           add.w   d4,d3
   68:           addq.w  #1,d2
   69: DDAS_00:
   70:           dbra    d0,DDA_loop
   71:
   72: DX_neg:
   73:           tst.w   X_neg
   74:           beq     DY_neg
   75:           neg.l   d1
   76: DY_neg:
   77:           tst.w   Y_neg
   78:           beq     DDA_exg
   79:           neg.l   d2
   80: DDA_exg
   81:           tst.w   Y_X
   82:           beq     DDA_quit
   83:           exg.l   d1,d2
   84: DDA_quit:
   85:           sub.w   d1,0(a2)
   86:           sub.w   d2,2(a2)
   87:           movem.l (sp)+,d0/d2-d5
   88:           rts
   89:
   90: X_neg:    .ds.w   1
   91: Y_neg:    .ds.w   1
   92: Y_X:      .ds.w   1
```

▶メモリ2Mバイトはやはり悲しい。　　吉田　勝浩(22)埼玉県

ん。リスト5は座標1から座標2までの最小移動量を出すプログラムです。最小移動量というのはX移動量もしくはY移動量が1であったとき，もう片方の移動量がいくつになるかを固定小数点で返すということです。

プログラムの注意点等はリスト4とほぼ同じですが，移動比を出すプログラムなので，角度が45°などのXYの移動量が同じ場合に限って処理を特別扱いして軽減しています。

また，31行目で検出する精度のセットをしています。これはそのままDDAのループ回数になるので，数字が多いほど精度はよいが処理が重くなり，少ないほど精度は悪くなるが処理が軽くなります。最大65535まで指定でき，1上がるごとに角度の分解能が倍になります。

たとえば10を指定した場合は0.088°まで検索することができます。極端な数値を指定するとものすごく遅くなるので，せいぜい15くらいまでが実用範囲でしょう。

*　　*　　*

と，まぁ2つのサンプルを紹介したわけですが，具体的にどのように使うのか一応参考例として書いておきましょう。

たとえばリスト4。これは全体的な処理を分散したいときや，正確な方向を検索するとき使用するのに向いていると思います。ループの内部が軽いのですが，毎回処理が必要なのでトータルでの処理は多くなります。

しかし，毎回の処理負荷が安定しているので，リアルタイムゲーム向きだと思います。うまく使えばシューティングの追尾弾などにも使えます。座標1を目標に応じて変えればいいのです。

そしてリスト5は，移動比を一度に計算してしまうので直線的な動きの管理などに向いていると思います。精度がコントロールできるため，敵弾が自機に向かうときにゲームレベルの違いによって精度を低くしたり高くしたりできるわけです。ただし，処理の高速化のため，精度を計算する掛け算をアセンブラに任せているので，精度によっていくつかルーチンを作らねばなりません（もっともX68030などのように掛け算の早い機種に限定すればルーチン内で計算してもいいんだけどね）。

◆ あとがき

今回セオリー集の第1弾として2つほど簡単なアルゴリズムの紹介などをしたわけですが，どうでしょうか。たまにはこういうお決まりごともいいんじゃないかと思います。今後，セオリー集とテクニック集を適当に織り交ぜてやっていこうかなぁと思っています。セオリー集といっても決して手を抜いているわけじゃありません。今後発展していく（?）と思われるゲームマネージメントシステムの重要なインタフェイス部分なのです。

そういえば，なんか今回は文章が重い感じがするなぁ。久しぶりの原稿でテンションが上がりきらなかった感じですね。次の記事はもっと軽快にいきましょう。

で，次はなにをやるかまだ確定していません。おそらくBGの応用編か，早速助っ人君参上“期待のスプライト編”かもしれません。今回のDDAを使用したパレット制御もいいなぁとも思っています。

どちらにしろゲームに実戦的に使用できるテクニックを供給していこうと思っていますので読者氏は自分のほしい技術が掲載されるまで気長に待っていてください（そのうち細かいテクニックを集めて番外編をやるかもしんない）。

それではこの連載が続くように私も勉強を進めますか。

参考文献

C言語による最新プログラム事典，技術評論社

協力　H.C.S.,Fill in Cafe' Co.,Ltd.

リスト5　DDA_2.s

```
==================  DDA_2.s  ==================
 1: *
 2: *        方向検索ルーチン    デジタル微分解析機アルゴリズム
 3: *                          ( Digital Differential Analyzer)
 4: *
 5: *        座標1から座標2までの最小XY移動比を出す
 6: *
 7: *        引数    a1        = 座標1のポインタ
 8: *                    X1.w
 9: *                    Y1.w
10: *               a2        = 座標2のポインタ
11: *                    X2.w
12: *                    Y2.w
13: *        返り値  d1 = Xの移動量    (上位16ビット整数値
14: *                d2 = Yの移動量     下位16ビット小数値)
15: *
16:
17:          .include        doscall.mac
18:
19:          lea.l   POINT1,a1
20:          lea.l   POINT2,a2
21:          jsr     DDA_2
22:
23:          DOS     _EXIT
24:
25: *                 X   Y
26: POINT1   .dc.w   000,000
27: POINT2   .dc.w   010,003
28:
29: *
30: DDA_2:
31: SEIDO    set     10        * 移動比の精度  (1~65535)
32:          movem.l d0/d3-d7,-(sp)
33:          move.w  0(a2),d4          * X座標をレジスタに読み込む
34:          sub.w   0(a1),d4          * X2-X1
35:          move.w  2(a2),d5          * Y座標をレジスタに読み込む
36:          sub.w   2(a1),d5          * Y2-Y1
37:
38:          move.w  #0,X_neg          * 判定ワークの初期化
39:          move.w  #0,Y_neg
40:          move.w  #0,Y_X
41:
42: XN_C:    tst.w   d4
43:          bpl     YN_C
44:          neg.w   d4
45:          move.w  #1,X_neg
46: YN_C:    tst.w   d5
47:          bpl     XY_C
48:          neg.w   d5
49:          move.w  #1,Y_neg
50: XY_C:
51:          cmp.w   d4,d5
52:          bne     XY_tst            * XとYがおなじ数値だった場合
53: notDDA:                            * 45°とみなして特別扱する
54:          move.l  #$10000,d1
55:          move.l  d1,d2
56:          bra     DX_neg
57: XY_tst:  bcs     DDA_start
58:          move.w  #1,Y_X
59:          exg.l   d4,d5
60:
61:          * ここがDDAの本体
62: DDA_start:
63:          clr.l   d2
64:          move.l  #$10000/SEIDO,d3
65:          move.w  #SEIDO-1,d0
66:          move.w  d4,d6
67:          lsr.w   #1,d6
68: DDA_loop:
69:          sub.w   d5,d6
70:          bpl     DDAS_00
71:          add.w   d4,d6
72:          add.w   d3,d2
73: DDAS_00:
74:          dbra    d0,DDA_loop
75:
76:          move.l  #$10000,d1
77:
78: DX_neg:
79:          tst.w   X_neg
80:          beq     DY_neg
81:          neg.l   d1
82: DY_neg:
83:          tst.w   Y_neg
84:          beq     DDA_exg
85:          neg.l   d2
86: DDA_exg
87:          tst.w   Y_X
88:          beq     DDA_quit
89:          exg.l   d1,d2
90: DDA_quit:
91:          movem.l (sp)+,d0/d3-d7
92:          rts
93:
94: X_neg:   .ds.w   1
95: Y_neg:   .ds.w   1
96: Y_X:     .ds.w   1
```

ローテク工作実験室　第3回

CRT分配器の製作

Taki Yasushi 瀧　康史

今回作成するのは標準CRTの機能を高解像ディスプレイとテレビに分配するためのCRT分配器です。プリントパターンつきですので，利用できる方は利用してください。あまり需要はないような気もしますが……。

発端

ことの発端は先週の特集でメガディスプレイ計画を立てたときのことです。新しいモニタ(T660iJ)を購入したものの，これは水平30kHz～82kHz，垂直55Hz～90Hzという代物でした。

24kHzはX68000ではオマケのようなものだからさておき（といえるようになったのも，1024×848モードを使わなくて済むようになったから。詳細は先月号を参照のこと)，15kHzが映らないのはやっぱり痛いのです。いくらほとんどのゲームが31kHzで映るとはいえ……ねぇ。

私の場合，もともと使っていたCU-21HDは売ったわけではありませんし，電波新聞社のXAV-1s（15kHz画像をテレビに映し出すもの。出力としてビデオ出力，S出力がある）もあるので，とりあえずはモニタ切り替え機でパチパチと切り替えればそれでも構いません。

でもこれがまたちょっとうざったいのです。モニタだけでなく，パソコンも複数あるので，モニタ側の入力も切り替えねばならないからです。うちの場合，T660iJは自動で入力を切り替えられるからともかく，CU21の前には4対1のモニタ切り替え機がドカっと座っています。それと，もうひとつテレビがありますしね。これには，NEO・GEOやらPCエンジンやら，MEGA DRIVEが接続してありますから，ゲームをしやすい床の上に置いてあります。冬ならば，ホットカーペットが敷いてありますし。なによりも，多人数でゲームをするのにふさわしい場所に置いてあるってわけです。

そういった背景上，やっぱりX68000でも多人数でゲームをするときは，TVに映したくなります。画像のよし悪しは二の次です。だってアクションを多人数でするんだから，やっぱりやりやすいほうがいいですよね。

ここまで入出力が絡んでくると，当然切り替え機が増えてきちゃうので，すごく面倒くさい環境になってしまいます。そこで，お決まりのローテクで，なんとか手抜きができないか考えてみました。

よくよく考えれば，RGBモニタはデータが一方通行です。入力は自動にしても，手動にしても切り替える必要がありますが，出力は切り替えるのではなく，すべてに同じ出力を送っても構わないことになります。

すなわち，ここにモニタ分配器を持ってきたらどうでしょうか？　愛機X68000 XVIの画像出力は，等しくすべてのモニタに出力されることになります。

先月のメガディスプレイの出力も，映るモニタにだけ映して，ゲームなどはほかのモニタで行います。長残光と短残光という組み合わせも面白いかな？（もっとも，長残光モニタは発色がよくないので，X68000にはおすすめできないけど）

2台のモニタを生かして，対戦ゲームに明け暮れることもできます。使用の仕方は自由自在。これぞローテクということで，今月はCRT分配器を作成することに決めました。

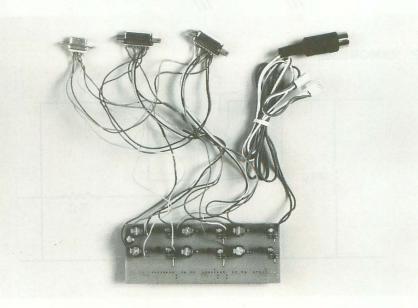
写真1

お決まりってことで

とりあえず，お決まりってことで，直接分配してみました。やっても絶対無駄とわかっていながらやってみますと，やっぱり駄目でした。理由は簡単で，出力を分けることにより，本来ひとつのモニタだけに出力されるように設計された信号（電流）が，2つに分かれて小さくなってしまうからです。

RGBモニタへの画面の出力は，RGBとH-SyncとV-Syncです。このうち，RGB信号はアナログ信号です。

予想では画面が暗くなるだけかな？　と思ったのですが，ちょっと予想外の行動をしてくれました。T660iJは画面の上のほうだけ明るいのですが，CU21HDは画面の明るさはあまり落ちずに映ります。

CU21HDとT660iJの入力インピーダン

スは，どちらも75Ωだと思うのですが，どうやら，CU21の入力インピーダンスはT660iJの入力インピーダンスよりも低いようです。インピーダンス＝抵抗ですから，抵抗が低いほうへ電流がたくさん流れているってわけですね。75Ωが並列になり37.5Ωになってしまうせいで，それぞれにかかる電圧も低くなってしまうという考え方のほうがわかりやすいかもしれませんね。

また，どちらも同期できる範囲で分配したところ，画面の輝度，色はおかしくなっても，同期がずれたりすることはありませんでした。つまりこれは，同期信号はどちらのディスプレイにもちゃんと分配されているということです。

同期信号の2つは，本体添付の取扱説明書を見ればわかるとおり，TTL出力です。TTLにはファンという概念があって，これは出力10ファンのICには，入力1ファンのICが10個まで並列につなぐことができますよという概念です。X68000側のH-SyncとV-SyncのファンアウトについてはOutside X68000，X68030Inside/Outで3機種の回路をチェックすることにしました。それぞれのマシンで少しずつ違ったものの，決まってTTLのインバータを通しているようなので，ファンアウトはそれなりにあるでしょう。

モニタ側の定格がわからないので，あまり強いことはいえませんが，おそらくこの2つの信号は大丈夫だと思います。普通のTTLはICを2，3個ぶら下げたところでどうってことないはずですが，どうしても心配な人は分配器にバッファICでもつけるとよいかもしれません。そうすれば，負荷がかかって壊れる場合，壊れる場所は接続したバッファICと相場が決まりますから，修理も簡単です。どうせそんなにたくさんのモニタを同時につなげないでしょうし，壊れたら私はX68030を買うつもりなので，ここは，放っておくことにします。

図1　基本回路図

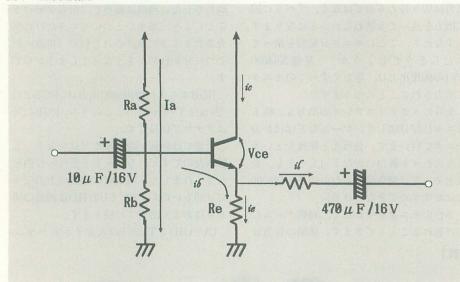
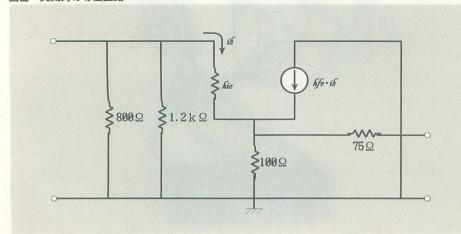



図2　交流時の等価回路

コレクタ接地

同期信号はこのまま分配すればよいことがわかりましたが，RGB信号を分配するという根本的な部分が，依然として解決していません。

それにはまず，RGB信号にコレクタ接地をしたトランジスタ回路を接続する必要があります。コレクタ接地とは別名，エミッタフォロアともいわれ，簡単にいえば，インピーダンス変換回路です。

電圧増幅は1倍，すなわち電圧利得はまったくありません。電子回路などを勉強したときに，なぜ電圧増幅が1倍なものが必要なのか？　と思ったかもしれませんが，実はこういうところに利用するのです。

今回のケースでは，電圧，つまり信号の波をそのまま複数のモニタに送りたいというものです。したがって，電圧増幅は必要ではありません。しかし，コード結線で単純に分配するとモニタが必要とする分，電流が取れなくなってしまいます（電圧，つまり信号は，電流に乗って進んでいると考えるとわかりやすいかもしれません）。

ぶら下がった回路に合わせて電流を流そうとするが流すほど定格がないというわけで，出力する側のIC，ないしはトランジスタに負荷がかかり，結果，電圧効果が起こります。ひどい場合，なにか部品が壊れたりするので注意が必要です。普通はそんなに簡単には壊れないように作ってありますが，回路の定格に幅が持たせられないケースではどうなるかはわかりません。

そこでインピーダンス変換回路の出番です。この回路は電圧信号はそのまま伝えますが，電流網は入力と出力ではまったく独立しています。入力インピーダンスは設計者が好きなように決められますし，出力インピーダンスも好きなように決められます。

図1を見てください。これは今回利用する回路の基本部分です。説明するのにわかりづらいので，交流のときの等価回路として，図2を書いておきます。

入力は左側の端子から入り，電流は800Ω，1.2kΩ，hieと3つの抵抗へ，オームの法則に従って流れ込みます。入力インピーダンスは，800Ω，1.2kΩ，hieの3つの抵抗値の並列ですが，hieはこのケースでは大きくなりますし，今回は厳密な入力インピーダンスを調べたいわけではないので，ここは800Ωと1.2kΩの抵抗の並列だけで入力インピーダンスを計算します。すると，

800｜｜1200＝480

で入力インピーダンスは，480Ωとなります。要は75Ωを割らなければよいので，この回路を利用して分割した場合，6分配（これは(1/75)/(1/480)>6という計算に基づく）までならできることになります。しかし，実際はhieが入ると少し抵抗が小さくなるので5つぐらいにしておいたほうがよいかもしれません。3分配ぐらいできれば普通は足りるのではないでしょうか？

回路の原理

実際の回路を説明する前に，設計についてさらっと流してお話ししましょう。興味のある方は専門書をご覧ください。私自身，アナログ回路は苦手なほうなので，もっともっとよい方法があるに違いありません。いやあるでしょう。

まず，X68000の取扱説明書を見ます。アナログRGB信号出力用コネクタの一覧表を見ると，RGBの信号にはそれぞれアナログ0.7Vp-p(75Ω終端時)と記述されています。これはどういうことかといいますと，信号は0.7Vの振幅幅で揺れているということなので，0Vを中心にしたら±0.35Vで振幅しているということになります。

出力に関する部分は，X68000と同じほうがいろいろと都合がよいので，X68030に合わせて，図1のような出力にしました。エミッタ側に100Ω，出力にコンデンサを挟んで75Ωという部分です（これは出力インピーダンスを調整するため）。

それでは直流について考えることにします。コレクタからエミッタに流れる電流網はどうなっているのでしょう。まずオームの法則より，

Vcc＝Vce＋Re･Ie

が導き出されます。

Re，Ieは，エミッタについている抵抗と電流です。Vcc=5V，Re=100Ωというのがわかっている事実となります。

図1からわかるとおり，

ie＝ib＋ic

です。しかし，トランジスタの法則から，

ic＝hfe･ib

で，hfeは通常100以上ですから，この場合，ic＝ieと考えても構いません。したがって，上の式をVce-icの特性式で表すと，

ic＝-1/Re･Vce＋Vcc/Re

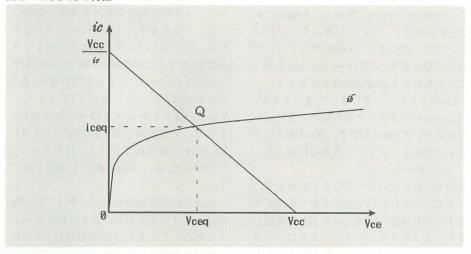
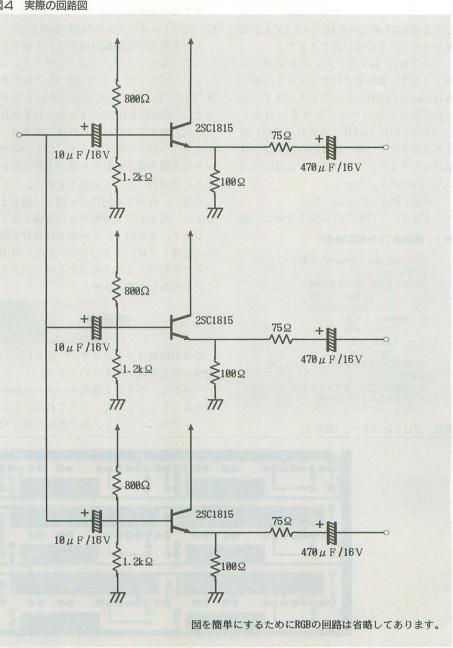
となるわけです。したがって，図3のようなグラフができあがります。

次に決めることは動作点です。

動作点Qはこのグラフ上でほぼ真ん中にいるのが得策です。この理由はibが図に書

図3　Vcc-icの特性

図4　実際の回路図

図を簡単にするためにRGBの回路は省略してあります。

き込まれているようにしなっているからです。今回，0.7Vp-pですから，振幅はたかが±0.35Vです。だいたい真ん中にいればよいので，動作点QはVccの半分ぐらいが妥当になります。したがってVceq=2.5V，これにあわせてiceq=25mAとなります

hfeはトランジスタによって違いますが，（これはデータブックを参照のこと）だいたいにおいて100以上です。仮に100ということにすると，トランジスタの特性より，

ic＝hfe・ib

なので，だいたいib＝250μAとなります。

さて，入力は0.7Vp-pですから交流です。こういった場合，0Vから±0.35Vに振幅します。よく考えるとトランジスタはベースからエミッタへの一方向しか流れませんから，-0.35Vになるときは電流を流すことができません。そのためには，0〜0.7Vまで電圧を持ち上げる必要が出てきます。

そのために，Iaを流すRa，Rbが必要になってきます。電圧を上げてしまうと，逆に入力のほうに流れていってしまいますから，それを防ぐためにコンデンサをつけます(10μFのもの)。IaはIbにも流れますが，回路を安定させるには，IaはIbの10倍以上は必要になります。よって，Ia＝10・Ib＝2.5mA。Vccは5Vですから，

(Ra＋Rb)＝Vcc/Ia＝2kΩ

になります。このRa，Rbの振り分けは，適当に1k，1kと決めてしまうのではなく（動きはしますが），自分がさっき決めた動作点に従って決めます。つまりVce＝2.5Vですから，Ve＝2.5V(トランジスタのエミッタの電圧)です。この点からVbe＝0.7V(トランジスタのベース，エミッタ間の電圧，0.7というのはだいたいにおいて)ですから，Vb=3.2V(トランジスタのベースの電圧)となり，そのためには，Rb:Ra=3.2:1.8とRa，Rbを計算すればよいことになります。したがって，Ra＝800Ω，Rb＝1.2kΩなのです。

余談ながら出力のコンデンサは，Ra，Rbで持ち上げた電圧がそのまま伝わらないように存在しています。75Ωの抵抗は，インピーダンス調整のためです。

さてこの回路は，入力インピーダンス480Ω，出力インピーダンス75Ωの回路です。

入力と出力の電流網は厳密にいえばつながっているのですが，ほぼつながっていないといってよいでしょう（Ib＋ic＝Ieであり，Ib＜＜Ic。Ie＞＞Ilであるため）。したがって，この回路をRGBの3つ分それぞれ作り，入力地点で3股に分ければ，出力はちゃんと1つひとつ出てきます。ひとつのモニタの電源を落としたら輝度が上がるとか，そういう間抜けなことはありません。

実際に，利用する回路図を図4に描きましょう。例として3分配のものを描くことにします。それ以下，それ以上は自分で勝手に想像して描いてください。また，RGBはそれぞれ同じものを利用するので，あえてひとつ分しか描きません。

表1　部品表（3分配の場合）

1．D-SUB15pinメスRGB用，4個
2．2SC1815，9個
3．コンデンサ　10μF/16V,9個
　　470μF/16V,9個
　　220μF/16V,1個
4．抵抗800Ω，9個
　　1.2kΩ，9個
　　100Ω，9個
　　75Ω，9個
5．基板。図5を焼き付けたプリント基板なら楽
6．接続ケーブル。おきまりですね
7．これらをしまうための箱

部品表

部品の種類は少ないですが，作る回路の規模に応じて数が増えていきます。

今回から，プリント基板のパターン図を載せることにしました。こうすれば，これをフィルムにコピーし，感光基板へ焼き付けるだけで同じものができますからね。

表1の1から説明していきましょう。

まず，15ピンコネクタ，とりあえず私は，普通の15ピンRGBと，DOS/V機などでよく利用されるミニ15ピンRGBがほしかったので，2股でも違うものにしました。

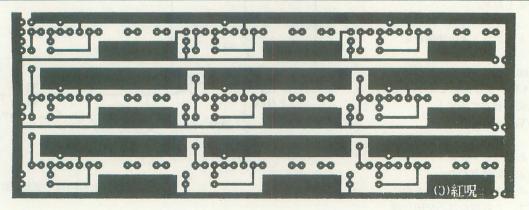
モニタケーブルはオスなので，メスを買ってきましょう。忘れていましたが，真ん中にアイテムが増えるのですから，RGBケーブルを余計に1本買ってくることを忘れずに，いままでX68000→RGBモニタというところが，X68000→CRT分配器→RGBモニタになるわけですから，必ず1本，余計にいるはずですよね。ちなみに，普通の15ピンはD-SUB15ピン（メス）です。

2は心臓部のトランジスタです。2SC1815という使いやすいポピュラーなものなので，これを機会に200個ぐらい買っておくとなにかよいことがあるかもしれません。さすがにちょっと買いすぎだと思っていますが……（どうでもいいことですが，どうして秋葉原って，バラ，10個，100個，200個というまとまりが多いんでしょう。50個あたりが手頃だと思うんだけどなぁ。場所によってはバラでいきなり100個単位になっちゃうし。秋葉原だと1個5円の抵抗も，100個100円ということはザラなのよ）。

3はコンデンサです。上の2つはフィルタ用，220μFは電源を安定させるためのフィルタですから適当で構いません。

4は抵抗です。今回，画面に映さなければならないという関係上，ノイズが乗るのがいやだったんで，なんと金属皮膜抵抗を買ってきました。100個単位で800円ぐらいしましたが，効果は謎です。金属皮膜抵抗はノイズだけでなく，ばらつきも小さいので，RGB間でのばらつきが起こりにくいのではないでしょうか。RGBがばらついてしまったら，色が変わってしまいますからね。それはさすがに嫌でしょ？

5は基板です。図5のとおりプリント基

図5　プリントパターン(原寸大)

板を起こせば，簡単にできますが，相変わらずセンタポンチとドリルでの穴開けが面倒くさいです。試作段階ではガラエポも使っていたんですが，ガラエポ基板を利用するときに限って失敗してしまいました（ガラエポ基板は普通のものよりも2倍以上高い）。

6は接続ケーブル。これは，RGBコネクタと基板をつなぐものです。私は道具箱の中に落ちていたケーブルを利用しました。本当は，こういうところでいいものを使わないとノイズが乗っちゃうんですけどね。短いから大丈夫でしょう。多分……。

7は箱です。記事を書いてる現在ではまだ箱は作っていません（どうしてこう，いつも黙ってればばれないことをバラしてしまうのでしょう？）。

製作にあたって

基板は，プリント基板じゃないとやっぱりノイズが乗りやすいみたいです。試作段階で蛇の目基板に回路を乗せてみたのですがゴーストが結構目だってしまい，ちょっと嫌な状態になってしまいました。

できるならば，図5のとおりにフィルムにコピーし，感光基板を使ってエッチングしてください。そういう環境にいないとなかなかやるのは難しいと思いますが。

今回の回路は同じ回路をいくつも作るだけで，たいした注意はありません。しいていうならば，ノイズ対策をしっかりすることです。余計なコードは延ばさずに，きちっとまとめ上げてください。

電源は5Vですが，私は前回と同じくテレビコントロール端子から取りました。いろいろゴテゴテつけていくと，電源がいつか足りなくなるので，9Vコンセントでも作って，レギュレータでひとつずつ電源回路を内蔵してしまいたいところですが。大きな定格の9Vコンセントがあれば，9Vアダプタが一気に減りますからね。SC-55，CM-64，MEGADRIVE，MEGA-CD，電話機，モデム，AVセレクタ……9Vが必要なものって多いですからね。9Vコンセントを作ればひとつですんじゃいますし。

余談はさて置き，今回の回路製作は別に難しいところはなにもないので，その分，愛をこめて作ってあげてください。

使用感

基板剥き出しですが，とりあえず使ってみました。これでストIIダッシュが2台のモニタで対戦できるではないか！　って，そのために作ったのではありません。念のため。

RGB分配回路は別に特殊なことはしていないので，ゲーム機のRGBやPC-9801などにも使えるはずです。CRTには寿命があります。初期型，ACE時代に買ったユーザーはそろそろ寿命が近いのではないでしょうか？

そうだったら，先月のメガディスプレイも含めて，17インチのハイレゾモニタあたりがお買い得ですよね。そういった意味でこういうものを作ったのですが，いかがでしょうか？

使用したときに真っ先に感じたのは，やっぱりセレクタがひとつ少なくなったせいか，セレクタ相当品がついていることを意識せずに使えるのはたいへん便利だということです。

私の使っているT660iJは入力が2系統あり，フロントパネルで切り替えられるのですが，いま，電源がONになったものを優先して表示するという機能があるせいで，これもまた，なにも意識せずに使うためには重宝します。相当するセレクタがOutside X68000に掲載されているので作ってみてはいかがでしょうか？　セレクタがついていることをすっかり忘れてしまいますよ。

そうそうRGBケーブルを途中で出すのだから，AUX-INとAUX-OUTをつけてSC-55を真ん中に差せば，CZ-600シリーズのステレオスピーカでもMIDIの音を鳴らすことができますよね。

いろいろ工夫すると面白いことができそうですから，CRTコントローラということで，大きな箱を買ってくるのもいかがでしょうか？　そのうち，変な回路も載っけますので……。

まとめ

今月は回路は簡単な割に実は結構たいへんでした。動いてもゴーストが出たり，なんで動くか謎な状況で動いていたり……。

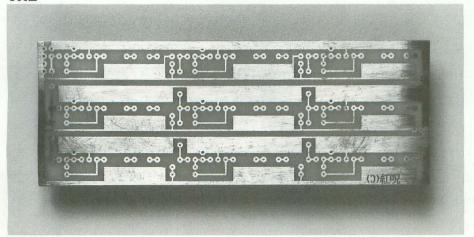
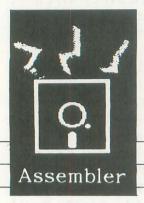
今回，初めて感光基板というものを使ってみました。私の友人が作ったZ_CADというツールを使って，パターンを仕上げ，IO735Xで印刷してフィルムコピー。現像液の温度が高すぎて最初の現像は失敗し，次はコピー機を利用したときに，トナーが薄かったせいで失敗。3度目の正直で，やっと写真のような基板を作成することができました。

え？　どうせプリント基板にするなら，WAVE BLASTER（WB）のやつがほしいですって？　あああ，しょうがない，来月覚えていたら，やりましょう。私もWBはまとめて回路にしたいと思ってましたから。どうせ，RS-MIDIの作り方もやったほうがいいかなーって思ってましたし。WBとWAVE BLASTERインタフェイス（WBI）だけで，MIDIできたら確かに便利ですからねぇ。個人的にはSC-55よりよい音だと思ってますし。

さて，来月は引き続きCRT関係のサポーター回路を作ろうかと思っていますが，予定は未定なのでどうなるのかは謎です。

それにしても，なにかのせいでむちゃくちゃ忙しいのです。遊びたいけど遊ぶ暇がありません。遊べないときに限って，自分の家に遊ぶものがたくさんあることに気がつくんですよね。まるでマーフィーの法則。

いつか，いま自分がやっていることが実ることを祈りながら，今月はこれでお別れということで。ではまた。

写真2

XMODEMによるファイル転送プログラム

Murata Toshiyuki 村田 敏幸

複数のパソコンがある場合，データのやりとりをしたいときがあります。共有できる大きなメディアがあれば問題ありません。しかし，そうでないときは……。今回はRS-232Cを使ったファイル転送です。プログラムに関する詳しい説明は次回となります。

今回はデータ伝送関係をつつき回すつもりだったのだが，諸事情により(こればっか)，背景や用語の説明は次回に送って，サンプルプログラムとその使い方だけを先に示しておくことにする。

RS-232C制御モジュール

まず，リスト1がRS-232C制御サブルーチン集。IOCSコールと同名/同機能のサブルーチンがひととおり用意されている(表1)。引数や戻り値もIOCSコールに揃えてあるので，

IOCS INP232C → jsr inp232c

のような単純な置き換えが可能だ。ただし，細部では以下の相違がある。

1） スーパーバイザモード専用

リスト1のサブルーチン群を利用する際には，事前にスーパーバイザモードに移行しておかなければならない。

2） 事前/事後処理が必要

リスト1のサブルーチン群は，IOCS SET232C相当の通信パラメータをd1に入れてサブルーチンopen232cを呼び出して初めて利用可能になる。ま

表1 RS-232C制御サブルーチン

```
●open232c：初期化
  入力 d1.w モード
        bit15,14：ストップビット長
                00：2
                01：1
                10：1.5
                11：2
        bit13,12：パリティ
                00：パリティなし
                01：奇数パリティ
                10：偶数パリティ
                11：パリティなし
        bit3～0：ボーレート
                0000：75
                0001：150
                0010：300
                0011：600
                0100：1200
                0101：2400
                0110：4800
                0111：9600
                1000：19200
                1001：38400
                1010：76800
  出力 d0.w FFFFH：エラー
●close232c：後始末
  入力 なし
  出力 なし
●set232c：通信パラメータの設定（IOCS SET232C）
  入力 d1.w モード（open232cに同じ）
  出力 d0.w 設定前のモード
●lof232c：入力バッファ内文字数取得（IOCS LOF232C）
  入力 なし
  出力 d0.l バッファ内文字数
●flush232c：入力バッファ内容破棄
  入力 なし
  出力 なし
●inp232c：1文字入力（IOCS INP232C）
  入力 なし
  出力 d0.w 入力文字/シグナル
        bit15  1：Break信号検出
        bit14  1：タイムアウト
        bit10  1：フレーミングエラー
        bit9   1：受信バッファオーバーラン
        bit8   1：パリティエラー
        bit7～0 入力文字コード
●isns232c：1文字先読み（IOCS ISNS232C）
  入力 なし
  出力 d0.l 入力文字/シグナル
        0001XXXXH：入力あり（XXXX=入力文字ほか）
        00000000H：入力なし
●out232c：1文字出力（IOCS OUT232C）
  入力 d1.b出力文字
  出力 なし
●osns232c：出力可能かどうかの検査（IOCS OSNS232C）
  入力 なし
  出力 d0.l 00000004H：出力可
            00000000H：出力不可
●starttimer：タイマ始動
  入力 d1.w 待ち時間（1/10単位）
  出力 なし
●stoptimer：タイマ停止
  入力 なし
  出力 なし
●conttimer：タイマ再開
  入力 なし
  出力 なし
```

た，使い終わったらサブルーチンclose232cを呼び出して後始末をする必要がある。なお，open232cを呼び出した時点でIOCSのRS-232C関連コール(ひいては，Human68kのAUX関係)は事実上無効になるので，混用はできない。

3）文字長8ビット専用

サブルーチンset232c(IOCS SET232C相当)の引数のうち，文字長の指定は無視される。また，IOCS SET232Cでは隠れ引数として，d1の第8ビットによりシフトイン/シフトアウト制御の有無が指定できるが，これも無視される。

4）フロー制御方式固定

リスト1はXON/XOFF制御するかどうかを，IOCSのようにSET232Cの引数で指定せずに，アセンブル時に決定するようになっている。掲載したリストでは，XON/XOFF制御する版が生成され，34行の代わりにコメントで殺してある33行を復活するとXON/XOFF制御しない版になる。また，35行を復活すると，IOCSではサポートされていないRS信号によるフロー制御を行う版が生成される[1]。

5）エラーの判別

サブルーチンinp232cやisns232c(それぞれ，IOCS INP232C，ISNS232C相当)は，d0の第8ビットにパリティエラーの有無を返す(IOCSではパリティエラーを起こした文字も何食わぬ顔でそのまま入力される)。また，第9ビットが受信バッファオーバーラン，第10ビットがフレーミングエラーを表す。

6）Break信号の検出

サブルーチンinp232c，isns232cは，Break信号を検出すると8000_Hを返す。

7）タイムアウト対応

d1.wに1/10秒単位の時間を入れてサブルーチンstarttimerを呼び出すと，タイマが作動し，指定時間後にタイムアウト(時間切れ)シグナルが内部的に生成される。タイムアウトはサブルーチンinp232c，isns232cの戻り値が4000_Hになることで示される。タイマはタイムアウトが発生するか，stoptimerが呼び出された時点で停止する。

8）高速伝送対応

IOCSでは9600bpsまでしかサポートしていない[2]のに対して，リスト1は19200/38400/76800bpsにも対応している。

XMODEMファイル転送プログラム

XMODEMはWard Christensenが考案した古典的なバイナリファイル転送プロトコルだ。大雑把にいうと，送信側はファイルを128バイト単位に分割し，ブロックの通し番号とチェックサムをつけた図1の形式で1ブロックずつ送る。受信側はブロック番号とチェックサムを確認して，問題なければACK(ASCII 06_H)で，誤りが見つかったらNAK(ASCII 15_H)で応える。送信側はそれを受けて，次のブロックを送るか，いま一度同じブロックを送るかを決める。すべての送信がすんだら，送信側はEOT(ASCII 04_H)を送り，受信側がACKで応えれば通信完了だ。

XMODEMには誤り検出方式をCRCにした変形や，ブロックサイズを1024バイトにした変形などがあり，さらに改良版のYMODEM, ZMODEMも存在する。今回作成したプログラムは，このうち，128バイトブロックのチェックサム方式とCRC方式のXMODEMにのみ対応している。

リスト2が共通ヘッダ，リスト3が送信側プログラム，リスト4が受信側プログラムとなっている。アセンブル/リンク手順はコメントに書いておいた。使用法だが，2台のX68000/030間のファイル転送に利用するのであれば，両者をRS-232Cクロスケーブルで接続し，SPEED.Xで通信パラメータの設定を合わせたうえで，まず，送信側，

A>TXMODEM 送信するファイル名

続いて，受信側，

A>RXMODEM 出力先ファイル名

の順に2つのプログラムを起動する。他機種とのファイル転送(当然，その機種用のXMODEMプログラムが別途必要)や，電話回線を通じての転送もこれに準じる。ここで，XMODEMでは，ファイル末の128バイトに満たない端数部分も必要なだけのEOF(ASCII $1A_H$)を補って128バイトブロックにして転送することになっており，ファイルサイズについての情報は通信相手に伝えられない。リスト3は最終ブロック末尾のEOFをすべて削除するので，ファイル末尾にEOFがあると勝手に削られることになる。使用時にはこの点に注意してほしい。

なお，リスト3，4をアセンブルする前に，リスト2の5行を復活しておくと76800bps専用になる。X68000/030間のファイル転送にのみ利用するのであれば，この変更を加えておいたほうがよいだろう。

では，続きは次回。

1）RSDRV.SYSはRS制御にも対応している。

2）RSDRV.SYSは19200bpsにも対応している。

図1 XMODEMのデータブロック

リスト1　232C.S

```
  1: *          RS-232C制御モジュール(非同期通信用)
  2:
  3:                  .include        doscall.mac
  4:                  .include        iocscall.mac
  5: *
  6:                  .xdef   open232c
  7:                  .xdef   close232c
  8:                  .xdef   set232c         *SET232C
  9:                  .xdef   lof232c         *LOF232C
 10:                  .xdef   isns232c        *ISNS232C
 11:                  .xdef   inp232c         *INP232C
 12:                  .xdef   osns232c        *OSNS232C
 13:                  .xdef   out232c         *OUT232C
 14:                  .xdef   flush232c
 15:                  .xdef   starttimer
 16:                  .xdef   conttimer
 17:                  .xdef   stoptimer
 18: *
 19: NMAXCHR          equ     2048            *入力バッファサイズ
 20: *
 21: NONE             equ     0
 22: SOFT             equ     1
 23: HARD             equ     2
 24: *
 25: .ifdef  XMODEM
 26: FLOWCTRL         equ     NONE
 27: .ifndef IGNERROR
 28: IGNERROR         equ     0
 29: .endif
 30: .endif
 31: *
 32: .ifndef FLOWCTRL
 33: *FLOWCTRL        equ     NONE            *フロー制御しない
 34: FLOWCTRL         equ     SOFT            *XON/XOFF制御する
 35: *FLOWCTRL        equ     HARD            *RS制御する
 36: .endif
 37: *
 38: IPAUSE_BIT       equ     0
 39: OPAUSE_BIT       equ     2               *(= RR0のTxバッファ空ビット)
 40: IPAUSE           equ     1<<IPAUSE_BIT   *入力一時停止中フラグ
 41: OPAUSE           equ     1<<OPAUSE_BIT   *出力一時停止中フラグ
 42: *
 43: XON              equ     'Q'-'@'         *XONコード
 44: XOFF             equ     'S'-'@'         *XOFFコード
 45: *
 46: RSCTRL_BIT       equ     1               *WR5 RS制御ビット
 47: ERCTRL_BIT       equ     7               *WR5 ER制御ビット
 48: SENDBREAK_BIT    equ     4               *WR5 Break送出ビット
 49: RSCTRL           equ     1<<RSCTRL_BIT
 50: ERCTRL           equ     1<<ERCTRL_BIT
 51: SENDBREAK        equ     1<<SENDBREAK_BIT
 52: *
 53: .ifdef  XMODEM
 54: BREAKSIG         equ     $0018           *Break検出シグナル = CAN
 55: .else
 56: BREAKSIG         equ     $8000           *Break検出シグナル
 57: .endif
 58: TIMEOUTSIG       equ     $4000           *タイムアウトシグナル
 59: *
 60: SCCA_CMD         equ     $e98005         *SCC ch.A コマンドポート
 61: SCCA_DAT         equ     $e98007         *SCC ch.A データポート
 62: MFP_IERA         equ     $e88007         *MFP 割り込みｲﾈｰﾌﾞﾙﾚｼﾞｽﾀA
 63: TIMERA_BIT       equ     6               *IERAのTimer-A(V-DISP)対応ビット
 64: *
 65: SCC_EI           equ     %00_111_000     *下位割り込みを受け付ける
 66: RST_ERRINT       equ     %00_110_000     *受信エラー割り込みリセット
 67: *RST_TxINT       equ     %00_101_000     *送信割り込みリセット
 68: *RST_RxINT       equ     %00_100_000     *受信割り込みリセット
 69: RST_ESINT        equ     %00_010_000     *E/S割り込みリセット
 70: *
 71: DI               macro                   *割り込み禁止マクロ
 72:                  ori.w   #$0700,sr
 73:                  .endm
 74: *
 75: EI               macro                   *割り込み許可マクロ
 76:                  andi.w  #$f8ff,sr
 77:                  .endm
 78: *
 79:                  .offset 0
 80: FLAG:            .ds.b   1               *一時停止フラグ
 81: VECFLAG:         .ds.b   1               *ベクタ待避済みフラグ
 82: WR3SAV:          .ds.b   1               *WR3 記憶用
 83: WR5SAV:          .ds.b   1               *WR5 記憶用
 84: DOWNCTR:         .ds.w   1               *タイマ用カウンタ
 85: MODE:            .ds.w   1               *モード
 86: LEFT:            .ds.w   1               *バッファ内文字数
 87: HEADPTR:         .ds.l   1               *バッファ内データ先頭
 88: TAILPTR:         .ds.l   1               *バッファ内データ末尾
 89: RxINTSAV:        .ds.l   1               *割り込みベクタ待避用
 90: ERRINTSAV:       .ds.l   1               *
 91: ESINTSAV:        .ds.l   1               *
 92: BUFFST:          .ds.w   NMAXCHR         *バッファ
 93: BUFFED:
 94: SIZEofWORK:
 95: *
 96:                  .text
 97:                  .even
 98: *
 99: *          初期化
100: *
101: open232c:
102:                  DI
103: SAVREGS          reg     d1/a0-a1
104:                  movem.l SAVREGS,-(sp)
105:
106:                  moveq.l #6,d1           *タイマ用に
107:                  lea.l   vdispint(pc),a1 *  V-DISP割り込みを
108:                  IOCS    _VDISPST        *  設定する
109:                  tst.l   d0              *
110:                  bne     openerror       *V-DISP割り込みが使用中
111:
112:                  bclr.b  #TIMERA_BIT,MFP_IERA  *V-DISP割り込み禁止
113:
114:                  lea.l   work+RxINTSAV,a0
115:                  move.w  #$5c,-(sp)      *書き換えるベクタを
116:                  DOS     _INTVCG         *  待避する
117:                  move.l  d0,(a0)+        *RxINTSAV
118:                  move.w  #$5e,(sp)       *
119:                  DOS     _INTVCG         *
120:                  move.l  d0,(a0)+        *ERRINTSAV
121:                  move.w  #$5a,(sp)       *
122:                  DOS     _INTVCG         *
123:                  move.l  d0,(a0)+        *ESINTSAV
124: *                addq.l  #2,sp
125:                  st.b    VECFLAG-RxINTSAV-4*3(a0)
126:                                          *ベクタ待避済みの印
127:
128:                  pea.l   recieveint(pc)  *受信割り込みの
129:                  move.w  #$5c,-(sp)      *  ベクタを設定する
130:                  DOS     _INTVCS         *
131: *                addq.l  #6,sp           *
132:
133:                  pea.l   errorint(pc)    *受信エラー割り込みの
134:                  move.w  #$5e,-(sp)      *  ベクタを設定する
135:                  DOS     _INTVCS         *
136: *                addq.l  #6,sp           *
137:
138:                  pea.l   extstatint(pc)  *E/S割り込みの
139:                  move.w  #$5a,-(sp)      *  ベクタを設定する
140:                  DOS     _INTVCS         *
141:                  lea.l   2+6+6+6(sp),sp  *
142:
143:                  move.w  2(sp),d1        *d1 = 伝送パラメータ
144:                  bra     set232c0
145:
146: *
147: *          伝送パラメータ設定
148: *
149: baudtable:       .dc.w   $0821           *75
150:                  .dc.w   $0410           *150
151:                  .dc.w   $0207           *300
152:                  .dc.w   $0102           *600
153:                  .dc.w   $0080           *1200
154:                  .dc.w   $003f           *2400
155:                  .dc.w   $001f           *4800
156:                  .dc.w   $000e           *9600
157:                  .dc.w   $0006           *19200
158:                  .dc.w   $0002           *38400
159:                  .dc.w   $0000           *76800
160: *
161: set232c:
162:                  DI
163: SAVREGS          reg     d1/a0-a1
164:                  movem.l SAVREGS,-(sp)
165:
166: set232c0:        cmpi.w  #-1,d1          *モード取得?
167:                  beq     get232c         *
168:
169:                  andi.b  #$7f,d1         *RSDRV.SYS対策
170:                  cmpi.b  #11,d1          *無効なボーレートが指定されたら
171:                  bcs     setbaudrate     *  1200にする
172:                  move.b  #4,d1           *
173: setbaudrate:     moveq.l #0,d0           *ボーレートに応じた
174:                  move.b  d1,d0           *  BRG用カウンタ値を
175:                  add.w   d0,d0           *  テーブルから曳く
176:                  lea.l   baudtable(pc,d0),a0
177:
178:                  moveq.l #0,d0
179:                  lea.l   SCCA_CMD,a1
180:                  move.b  #3,(a1)         *WR3  bit0=0...受信動作停止
181:                  move.b  d0,(a1)         *
182:                  move.b  #5,(a1)         *WR5  bit3=0...送信動作停止
183:                  move.b  d0,(a1)         *
184:                  move.b  #14,(a1)        *WR14 bit0=0...BRG停止
185:                  move.b  #%00000010,(a1) *
186:                  move.b  #13,(a1)        *BRG用カウンタ上位バイト設定
187:                  move.b  (a0)+,(a1)      *
188:                  move.b  #12,(a1)        *BRG用カウンタ下位バイト設定
189:                  move.b  (a0)+,(a1)      *
190:
191:                  lea.l   work,a0         *受信キュー初期化
192:                  move.b  d0,(a0)         *FLAG = 0
193:                  move.w  d0,LEFT(a0)     *LEFT = 0
194:                  move.l  #work+BUFFST,d0 *
195:                  move.l  d0,HEADPTR(a0)  *
196:                  move.l  d0,TAILPTR(a0)  *
197:
198:                  move.w  d1,-(sp)        *WR4 bit7,6=01...x16ｸﾛｯｸﾓｰﾄﾞ
199:                  moveq.l #%00001111,d0   *    bit3,2=      ｽﾄｯﾌﾟﾋﾞｯﾄ長
200:                  rol.w   #4,d1           *    bit1  =      パリティ偶/奇
201:                  and.b   d1,d0           *    bit0  =      パリティ有/無
202:                  andi.b  #%00001100,d1   *
203:                  bne     setWR4          *
204:                  ori.b   #%00001100,d0   *
205: setWR4:          ori.b   #%01000000,d0   *
206:                  move.b  #4,(a1)         *
207:                  move.b  d0,(a1)         *
208:                  move.w  (sp)+,d1        *
209:
210:                  move.b  #15,(a1)        *WR15 bit7  =1 ...Break検出で
211:                  move.b  #%10000000,(a1) *                  E/S割り込み
212:
213:                  move.b  #1,(a1)         *WR1 bit4,3=10...ｷｬﾗｸﾀごとに
214:                  move.b  #%00010101,(a1) *                  受信割り込み
215:                                          *    bit2 =1...ﾊﾟﾘﾃｨｴﾗｰで
216:                                          *             受信エラー割り込み
217:                                          *    bit1 =0...送信割り込み禁止
218:                                          *    bit0 =1...E/S割り込み許可
219:
220:                  move.b  #3,(a1)         *WR3 bit7,6=11...受信8ビット
221: .if     FLOWCTRL.eq.HARD                 *    bit5  =1 ...オートモード
222:                  moveq.l #%11100001,d0   *    bit0  =1 ...受信動作許可
223: .else                                    *
224:                  moveq.l #%11000001,d0   *
225: .endif                                   *
226:                  move.b  d0,(a1)         *
227:                  move.b  d0,WR3SAV(a0)   *
228:
```

▶大学に入って遊べると思ったのに，化学実験なんて科目があって，毎週レポートに追われています。化学はほとんどわからないので，かなり大変です。ちなみに私の所属している学科は応用物理学科です……。　　坂江　直樹(19)神奈川県

```
229:                move.b  #5,(a1)            *WR5  bit7  =1 ...ERアクティブ
230:                moveq.l #%11101010,d0      *     bit6,5=11...送信8ビット
231:                move.b  d0,(a1)            *     bit3  =1 ...送信動作許可
232:                move.b  d0,WR5SAV(a0)      *     bit1  =1 ...RSアクティブ
233:
234:                move.b  #14,(a1)           *WR14 bit0  =1 ...BRG動作許可
235:                move.b  #%00000011,(a1)    *
236:
237:                move.b  #RST_ESINT,(a1)          *E/S割り込みリセット
238:                move.b  #RST_ERRINT,(a1)         *エラーリセット
239:                move.b  #SCC_EI,(a1)             *割り込み許可
240:
241:                moveq.l #0,d0              *
242:                move.w  MODE(a0),d0        *d0 = 前のモード
243:                move.w  d1,MODE(a0)        *モードを覚えておく
244:
245: setretn:       movem.l (sp)+,SAVREGS
246:                EI
247:                rts
248: *
249: get232c:       moveq.l #0,d0              *
250:                move.w  work+MODE,d0       *d0 = 現在のモード
251:                bra     setretn
252: *
253: openerror:     moveq.l #-1,d0             *初期化失敗
254:                bra     setretn            *
255:
256: *
257: *          後始末
258: *
259: close232c:
260:                DI
261: SAVREGS        reg     d0-d1/a0-a1
262:                movem.l SAVREGS,-(sp)
263:
264:                lea.l   work+RxINTSAV,a0
265:                tst.b   VECFLAG-RxINTSAV(a0)
266:                beq     resetscc
267:
268:                move.l  (a0)+,-(sp)        *RxINTSAV
269:                move.w  #$5c,-(sp)
270:                DOS     _INTVCS
271: *              addq.l  #6,sp
272:
273:                move.l  (a0)+,-(sp)        *ERRINTSAV
274:                move.w  #$5e,-(sp)
275:                DOS     _INTVCS
276: *              addq.l  #6,sp
277:
278:                move.l  (a0)+,-(sp)        *ESINTSAV
279:                move.w  #$5a,-(sp)
280:                DOS     _INTVCS
281:                lea.l   6+6+6(sp),sp
282:
283:                sf.b    VECFLAG-RxINTSAV-4*3(a0)
284:                                           *ベクタ未待避の印
285:
286: resetscc:      lea.l   SCCA_CMD,a0
287:                move.b  #15,(a0)           *E/S割り込みを禁止する
288:                sf.b    (a0)               *
289:                move.b  #1,(a0)            *割り込み関係リセット
290:                move.b  #%00010000,(a0)    *
291:                move.b  #3,(a0)            *受信器関係リセット
292:                move.b  #%11000001,(a0)    *
293:                move.b  #5,(a0)            *送信器関係リセット
294:                move.b  #%11101010,(a0)    *
295:
296:                suba.l  a1,a1              *V-DISP割り込み停止
297:                IOCS    _VDISPST           *
298:
299:                movem.l (sp)+,SAVREGS
300:                EI
301:                rts
302:
303: *
304: *          受信バッファ内文字数を得る
305: *
306: lof232c:
307:                moveq.l #0,d0
308:                move.w  work+LEFT,d0
309:                rts
310:
311: *
312: *          受信バッファを空にする
313: *
314: flush232c:
315:                DI
316:                clr.w   work+LEFT
317:                move.l  #work+BUFFST,work+HEADPTR
318:                move.l  #work+BUFFST,work+TAILPTR
319:                EI
320:                rts
321:
322: *
323: *          1文字入力(入力があるまで待つ)
324: *
325: *          out d0  bit 15   ... Break信号検出
326: *                  bit 14   ... タイムアウト
327: *                  bit 10   ... フレーミングエラー
328: *                  bit  9   ... 受信オーバーラン
329: *                  bit  8   ... パリティエラー
330: *                  bit 7~0 ... 受信した文字
331: *
332: inp232c:
333: SAVREGS        reg     a0-a1
334:                movem.l SAVREGS,-(sp)
335:
336:                lea.l   work,a0
337: iwaitloop:     tst.w   LEFT(a0)           *バッファが空なら
338:                beq     iwaitloop          *  入力があるまで待つ
339:
340: input:         moveq.l #1,d0              *バッファ先頭1文字を
341:                swap.w  d0                 *  取り出す
342:                movea.l HEADPTR(a0),a1     *
343:                move.w  (a1)+,d0           *
344:                cmpa.l  #work+BUFFED,a1    *
345:                bcs     updatehead         *
346:                lea.l   BUFFST(a0),a1      *
347: updatehead:    move.l  a1,HEADPTR(a0)     *読み込みポインタを更新する
348:                subq.w  #1,LEFT(a0)        *バッファ内文字数を更新する
349:
350: .if     FLOWCTRL
351:                cmpi.w  #NMAXCHR/4,LEFT(a0)        *バッファ使用量が
352:                bcc     inpretn                    *  1/4未満になったら
353:                bclr.b  #IPAUSE_BIT,(a0)           *  相手に出力を
354:                beq     inpretn                    *  再開させる
355: .if     FLOWCTRL.eq.HARD
356:                ori.b   #RSCTRL,WR5SAV(a0)         *RSをアクティブにする
357:                lea.l   SCCA_CMD,a1                *
358:                DI                                 *
359:                move.b  #5,(a1)                    *
360:                move.b  WR5SAV(a0),(a1)            *
361:                EI                                 *
362: .elseif FLOWCTRL.eq.SOFT
363:                moveq.l #XON,d1                    *XONを出力する
364:                bsr     out232c                    *
365: .endif
366: .endif
367: inpretn:       movem.l (sp)+,SAVREGS
368:
369:                rts
370:
371: *
372: *          1文字先読み
373: *
374: isns232c:
375:                moveq.l #0,d0
376:                tst.w   work+LEFT          *バッファにデータはある?
377:                beq     isnsretn           *バッファが空だった
378:
379:                move.l  a0,-(sp)           *{
380:                moveq.l #1,d0              *バッファ先頭1文字を
381:                swap.w  d0                 *  取り出す
382:                movea.l work+HEADPTR,a0    *
383:                move.w  (a0),d0            *
384:                movea.l (sp)+,a0           *}
385: isnsretn:      rts
386:
387: *
388: *          出力可能かどうか調べる
389: *
390: osns232c:
391: .if     FLOWCTRL.eq.SOFT
392:                moveq.l #0,d0
393:                btst.b  #OPAUSE_BIT,work+FLAG
394:                bne     osnsretn           *XOFF状態
395: .endif
396:                moveq.l #%00000100,d0      *RR0 bit2=1...出力可
397:                and.b   SCCA_CMD,d0        *
398: osnsretn:      rts
399:
400: *
401: *          1文字出力
402: *
403: out232c:
404: .if     FLOWCTRL.eq.SOFT
405: owaitloop1:    btst.b  #OPAUSE_BIT,work+FLAG      *出力一時停止フラグが
406:                bne     owaitloop1                 *  寝るまで待つ
407: .endif
408: owaitloop2:    btst.b  #2,SCCA_CMD        *RR0 bit2=1...出力可
409:                beq     owaitloop2         *送信バッファが空くまで待つ
410:
411:                move.b  d1,SCCA_DAT        *出力する
412:                rts
413:
414: *
415: *          タイマスタート/ストップ
416: *
417: starttimer:
418:                move.w  d1,work+DOWNCTR
419: conttimer:     bset.b  #TIMERA_BIT,MFP_IERA       *V-DISP割り込み許可
420:                rts
421: *
422: stoptimer:
423:                bclr.b  #TIMERA_BIT,MFP_IERA       *V-DISP割り込み禁止
424:                rts
425:
426: *
427: *          垂直同期割り込みハンドラ
428: *
429: vdispint:
430:                subq.w  #1,work+DOWNCTR
431:                beq     timeout
432:                rte
433: *
434: timeout:
435:                DI
436: SAVREGS        reg     d1/a0-a1
437:                movem.l SAVREGS,-(sp)
438:
439:                bclr.b  #TIMERA_BIT,MFP_IERA       *V-DISP割り込み禁止
440:                move.w  #TIMEOUTSIG,d1
441:                bra     rxintent
442:
443: *
444: *          外部ステータス変化割り込みハンドラ
445: *
446: extstatint:
447:                DI
448: SAVREGS        reg     d1/a0-a1
449:                movem.l SAVREGS,-(sp)
450:
451:                lea.l   SCCA_CMD,a1
452:                move.b  (a1),d1            *RR0を選択
453:                move.b  (a1),d1            *RR0 bit7=1...break信号
454:                bmi     rxbreak            *
455:
456:                move.b  #RST_ESINT,(a1)    *E/S割り込みリセット
457:                move.b  #SCC_EI,(a1)       *SCC割り込み再許可
458:                movem.l (sp)+,SAVREGS
459:                rte
460: *
```

▶ビジネスショウに行ってきました。シャープのブースにはやっぱり1台もX68000はない。ちっとは使ってくれてもいいのに……。　中村　洋一(27)神奈川県

```
461: rxbreak:       move.b  #RST_ESINT,(a1)  *E/S割り込みリセット
462:                move.b  SCCA_DAT-SCCA_CMD(a1),d1         *空読み
463:
464:                move.w  #BREAKSIG,d1     *Break検出シグナル
465:                bra     rxintent
466:
467: *
468: *          受信エラー割り込みハンドラ
469: *
470: errorint:
471: .ifdef  IGNERROR
472:                tst.b   SCCA_DAT                 *受信バッファ空読み
473:                move.b  #RST_ERRINT,SCCA_CMD     *エラーリセット
474:                move.b  #SCC_EI,SCCA_CMD         *SCC割り込み再許可
475:                rte
476: .else
477:                DI
478: SAVREGS        reg     d1/a0-a1
479:                movem.l SAVREGS,-(sp)
480:
481:                lea.l   SCCA_CMD,a1      *割り込み要因をd1.bに得る
482:                move.b  #1,(a1)          *
483:                moveq.l #0,d1            *
484:                move.b  (a1),d1          *
485:                lsl.w   #4,d1            *d1.wの上位バイト = 割り込み要因
486:                move.b  SCCA_DAT-SCCA_CMD(a1),d1
487:                                         *受信バッファを空読みする
488:                move.b  #RST_ERRINT,(a1)  *エラーリセット
489:                bra     rxintent
490: .endif
491:
492: *
493: *          受信割り込みハンドラ
494: *
495: recieveint:
496:                DI
497: SAVREGS        reg     d1/a0-a1
498:                movem.l SAVREGS,-(sp)
499:
500:                moveq.l #0,d1            *受信データを取り込む
501:                move.b  SCCA_DAT,d1      *
502:
503: rxintent:      lea.l   work,a0
504:
505: .if     FLOWCTRL.eq.SOFT
506:                cmpi.b  #XOFF,d1
507:                beq     xon                      *XOFFを受信した
508:                cmpi.b  #XON,d1
509:                beq     xoff                     *XONを受信した
510: .endif
511:                cmpi.w  #NMAXCHR,LEFT(a0)
512:                bcc     full             *バッファが一杯だった
513:
514:                movea.l TAILPTR(a0),a1   *バッファに溜めておく
515:                move.w  d1,(a1)+         *
516:                cmpa.l  #work+BUFFED,a1  *
517:                bcs     updatetail       *
518:                lea.l   BUFFST(a0),a1    *
519: updatetail:    move.l  a1,TAILPTR(a0)   *書き込みポインタ更新
520:                addq.w  #1,LEFT(a0)      *
521:
522: full:
523: .if     FLOWCTRL
524:                cmpi.w  #(NMAXCHR*3)/4,LEFT(a0)  *バッファ使用量が
525:                bcs     rxintretn                *  3/4を越えたら
526:                bset.b  #IPAUSE_BIT,(a0)         *  相手に出力を
527:                bne     rxintretn                *  待ってもらう
528: .if     FLOWCTRL.eq.HARD
529:                lea.l   SCCA_CMD,a1              *RSを非アクティブにする
530:                move.b  #5,(a1)                  *
531:                andi.b  #.not.RSCTRL,WR5SAV(a0)  *
532:                move.b  WR5SAV(a0),(a1)          *
533: .elseif FLOWCTRL.eq.SOFT
534:                moveq.l #XOFF,d1                 *XOFFを出力する
535:                bsr     out232c                  *
536: .endif
537: .endif
538: rxintretn:     movem.l (sp)+,SAVREGS
539:                move.b  #SCC_EI,SCCA_CMD         *SCC割り込み再許可
540:                rte
541: *
542: xon:           ori.b   #OPAUSE,(a0)     *出力一時停止フラグを立てる
543:                bra     rxintretn
544: xoff:          andi.b  #.not.OPAUSE,(a0)        *         寝かせる
545:                bra     rxintretn
546: *
547:                .bss
548:                .even
549: *
550: work:          .ds.b   SIZEofWORK
551:
552:                .end
```

リスト2 XMODEM.H

```
 1: *           TXMODEM.S/RXMODEM.S 共用定数定義
 2:
 3: *SPEED      equ     %01_00_11_00_00001000   *19200
 4: *SPEED      equ     %01_00_11_00_00001001   *38400
 5: *SPEED      equ     %01_00_11_00_00001010   *76800
 6: *
 7: CRCPOLY     equ     %0001000000100001       *CRC生成多項式
 8: *
 9: BLOCKSIZE   equ     128                     *データ部バイト数
10: FRAMESIZE   equ     3+BLOCKSIZE+1           *チェックサムXMODEMフレームサイズ
11: FRAMESIZE_CRC equ   FRAMESIZE+1             *CRC XMODEMフレームサイズ
12: *
13: SOH         equu    $01     *^A     ブロック先頭
14: *STX        equ     $02     *^B
15: EOT         equ     $04     *^D     送信終了
16: ACK         equ     $06     *^F     肯定応答
17: NAK         equ     $15     *^U     否定応答
18: CAN         equ     $18     *^X     中断
19: PADCHR      equ     $1a     *^Z     パッド
20: *PADCHR     equ     $00     *NUL
21: *
22: RETRY_T     equ     10*10   *リトライ間隔10秒
23: RETRY_C     equ     10      *リトライ回数10回
24: TIMEOUTSIG  equ     $4000   *タイムアウト内部コード
```

リスト3 TXMODEM.S

```
 1: *        XMODEMファイル転送プログラム(送信側)
 2: *
 3: *        作成法  as txmodem
 4: *                as 232c -sXMODEM -ox232c
 5: *                lk txmodem 232c
 6:
 7:                 .include        doscall.mac
 8:                 .include        iocscall.mac
 9:                 .include        xmodem.h
10: *
11:                 .xref   open232c
12:                 .xref   close232c
13:                 .xref   inp232c
14:                 .xref   out232c
15:                 .xref   starttimer
16:                 .xref   stoptimer
17: *
18:                 .text
19:                 .even
20: *
21: ent:
22:                 lea.l   inisp,sp
23:
24:                 clr.w   -(sp)           *入力ファイルを開く
25:                 pea.l   1(a2)           *
26:                 DOS     _OPEN           *
27: *               addq.l  #6,sp           *
28:                 move.w  d0,d7           *
29:                 bmi     error           *
30:
31:                 bsr     crc_comp        *crc計算の前準備
32:
33:                 clr.l   -(sp)
34:                 DOS     _SUPER
35:
36:                 pea.l   abort(pc)       *中断時終了アドレス設定
37:                 move.w  #_CTRLVC,-(sp)  *
38:                 DOS     _INTVCS         *
39:                 move.w  #_ERRJVC,(sp)   *
40:                 DOS     _INTVCS         *
41: *               addq.l  #6,sp           *
42:
43: .ifdef  SPEED                                   *RS-232C初期化
44:                 move.w  #SPEED,d1               *
45: .else                                           *
46:                 moveq.l #-1,d1                  *
47:                 IOCS    _SET232C                *
48:                 move.w  d0,d1                   *
49: .endif                                          *
50:                 bsr     open232c                *
51:                 addq.w  #1,d0                   *d0=-1なら初期化失敗
52:                 beq     error                   *
53:
54:                 lea.l   chksum(pc),a6           *デフォルトはチェックサムXMODEM
55:                 move.w  #FRAMESIZE-1,d5         *
56:
57:                 move.w  #60*10,d1               *待時間60秒
58:                 bsr     starttimer              *
59: idleloop:       bsr     inp232c                 *NAKか'C'が送られるのを待つ
60:                 cmpi.w  #CAN,d0                 *CAN?
61:                 beq     cancel                  *
62:                 cmpi.w  #TIMEOUTSIG,d0          *タイムアウト?
63:                 beq     cancel                  *
64:                 cmpi.w  #NAK,d0                 *NAK?
65:                 beq     do                      *  そうならチェックサムXMODEM
66:                 cmpi.w  #'C',d0                 *'C'?
67:                 bne     idleloop                *  そうならXMODEM/CRC
68:
69:                 lea.l   crctable,a2             *XMODEM/CRCが利用可
70:                 lea.l   crc(pc),a6              *
71:                 move.w  #FRAMESIZE_CRC-1,d5
72:
73: do:             bsr     stoptimer
74:
75:                 lea.l   out232c,a3              *a3 = 1文字出力ルーチン
76:                 lea.l   buff,a4                 *a4 = バッファ
77:                 lea.l   BLOCKSIZE.w,a5          *a5 = ブロックサイズ
78:
79:                 move.b  #SOH,(a4)+              *ヘッダ部を初期化する
80:                 move.b  #$01,(a4)+              *
81:                 move.b  #$fe,(a4)+              *
82:
83:                 movem.l a4/a5,-(sp)             *DOS READ の引数
84:                 move.w  d7,-(sp)                *
```

▶妹に「ワープロ買うんだけど，なにがいいかな」と聞かれたとき，おもわずメモリ＋EGWordが頭のなかに浮かんでしまった。　大内　直己(20)千葉県

```
 85:              subq.l  #3,a4
 86: mainloop:    DOS     _READ               *1ブロック分読み込む
 87:              cmp.l   a5,d0               *128バイトある?
 88:              bcc     calcbcc             *
 89:
 90:                                   *128バイト未満
 91:              tst.w   d0                  *
 92:              beq     term                *ファイルが尽きた
 93:
 94:              move.w  a5,d1               *ブロックの残りを埋める
 95:              sub.w   d0,d1               *
 96:              lea.l   3(a4,a5),a0         *
 97:              suba.w  d1,a0               *
 98:              moveq.l #PADCHR,d0          *
 99:              subq.w  #1,d1               *
100: fillloop:    move.b  d0,(a0)+            *
101:              dbra    d1,fillloop         *
102:
103: calcbcc:     lea.l   3(a4),a0            *ブロックチェックコード
104:              move.w  #BLOCKSIZE-1,d6     *(チェックサム or CRC)を
105:              jsr     (a6)                *  計算する
106:
107:              moveq.l #RETRY_C-1,d2
108: output:      movea.l a4,a0               *1ブロック出力する
109:              move.w  d5,d6               *
110: outputloop:  move.b  (a0)+,d1            *
111:              jsr     (a3)                *out232c
112:              dbra    d6,outputloop       *
113:
114:              bsr     waitack             *ACK待ち
115:              dbpl    d2,output           *NAKなら再送する
116:              bmi     abort
117:
118: next:        addq.b  #1,1(a4)            *ブロックシーケンス番号を
119:              subq.b  #1,2(a4)            *  進める
120:              bra     mainloop
121:
122: term:        moveq.l #RETRY_C-1,d2
123: termloop:    moveq.l #EOT,d1             *送信終了コードを送る
124:              jsr     (a3)                *out232c
125:              lea.l   eotmes(pc),a1
126:              IOCS    _B_PRINT
127:              bsr     waitack             *ACK待ち
128:              dbpl    d2,termloop         *NAKならEOTを再送する
129:              bmi     abort
130:
131:              lea.l   crlfmes(pc),a1
132:              IOCS    _B_PRINT
133:              bra     exit
134:
135: cancel:      lea.l   canmes(pc),a1       *相手の都合で中断
136:              IOCS    _B_PRINT            *
137:              bra     exit                *
138:
139: abort:       moveq.l #CAN,d1             *自分の都合で中断
140:              bsr     out232c             *
141:
142: exit:        bsr     close232c
143:
144:              moveq.l #-1,d1              *RS-232C関係IOCSリセット
145:              IOCS    _SET232C            *
146:              move.w  d0,d1               *
147:              IOCS    _SET232C            *
148:
149: error:       DOS     _EXIT
150:
151: *
152: *        ACKを待つ
153: *
154: waitack:
155:              moveq.l #RETRY_T,d1
156:              bsr     starttimer
157: waitackloop: bsr     inp232c             *1文字入力
158:              cmpi.w  #CAN,d0             *CAN?
159:              beq     abort               *
160:              cmpi.w  #TIMEOUTSIG,d0      *タイムアウト?
161:              beq     nakretn             *  そうならNAK扱い
162:              cmpi.w  #ACK,d0             *ACK?
163:              beq     ackretn             *
164:              cmpi.w  #NAK,d0             *NAK?
165:              bne     waitackloop         *
166:
167: nakretn:     bsr     stoptimer
168:              moveq.l #'.',d1
169:              IOCS    _B_PUTC
170:              moveq.l #-1,d0              *N=1
171:              rts
172:
173: ackretn:     bsr     stoptimer
174:              moveq.l #'*',d1
175:              IOCS    _B_PUTC
176:              moveq.l #0,d0               *N=0
177:              rts
178:
179: *
180: *        8ビットチェックサムを求める
181: *
182: chksum:
183:              moveq.l #0,d0
184: chksumloop:  add.b   (a0)+,d0
185:              dbra    d6,chksumloop
186:
187:              move.b  d0,(a0)
188:              rts
189:
190: *
191: *        16ビットCRCを求める
192: *
193: crc:
194:              moveq.l #0,d0
195: crcloop:     ror.w   #8,d0               *つぎの8ビットを取り込む
196:              moveq.l #0,d1               *
197:              move.b  (a0)+,d1            *
198:              eor.b   d0,d1               *
199:
200:              add.w   d1,d1               *テーブルを引いて
201:              move.w  0(a2,d1.w),d1       *  中間結果を更新する
202:              clr.b   d0                  *
203:              eor.w   d1,d0               *
204:
205: crcnext:     dbra    d6,crcloop          *データが尽きるまで繰り返す
206:
207:              move.b  d0,d1
208:              lsr.w   #8,d0
209:              move.b  d0,(a0)+            *CRC上位バイト
210:              move.b  d1,(a0)+            *CRC下位バイト
211:              rts
212:
213: *
214: *        CRC計算用のテーブルを作成する
215: *
216: crc_comp:
217:              lea.l   crctable+256*2,a0   *a0 = テーブル末尾
218:              move.w  #CRCPOLY,d1         *d1 = 生成多項式
219:
220:              move.w  #256-1,d2
221: comploop1:   move.w  d2,d0               *d0 = 255, 254, 253, ..., 0
222:              lsl.w   #8,d0
223:
224:              moveq.l #8-1,d3
225: comploop2:   add.w   d0,d0               *中間結果を1ビット左シフトする
226:              bcc     compnext2           *第16ビットが1なら
227:              eor.w   d1,d0               *  定数とXORを取る
228: compnext2:   dbra    d3,comploop2        *8ビット分繰り返す
229:
230:              move.w  d0,-(a0)            *テーブルに登録する
231:              dbra    d2,comploop1        *すべてのバイト値について繰り返す
232:
233: compretn:    rts
234: *
235: canmes:      .dc.b   '[CAN]'
236: crlfmes:     .dc.b   13,10,0
237: eotmes:      .dc.b   '[EOT]',0
238: *
239:              .bss
240:              .even
241: *
242: crctable:    .ds.w   256                 *CRC計算用テーブル
243: buff:        .ds.b   FRAMESIZE_CRC       *入出力バッファ
244: *
245:              .stack
246:              .even
247: *
248:              .ds.l   4096
249: inisp:
250:              .end    ent
```

リスト4 RXMODEM.S

```
  1: *       XMODEMファイル転送プログラム(受信側)
  2: *
  3: *       作成法  as rxmodem
  4: *               as 232c -sXMODEM -ox232c
  5: *               lk rxmodem 232c
  6:
  7:         .include        doscall.mac
  8:         .include        iocscall.mac
  9:         .include        xmodem.h
 10: *
 11:         .xref   open232c
 12:         .xref   close232c
 13:         .xref   inp232c
 14:         .xref   out232c
 15:         .xref   starttimer
 16:         .xref   stoptimer
 17: *
 18:         .text
 19:         .even
 20: *
 21: ent:
 22:         lea.l   inisp,sp
 23:
 24:         move.w  #$0020,-(sp)        *出力ファイルを作成する
 25:         pea.l   1(a2)               *
 26:         DOS     _CREATE             *
 27: *       addq.l  #6,sp               *
 28:         move.w  d0,d7               *
 29:         bmi     error               *
 30:
 31:         bsr     crc_comp            *crc計算の前準備
 32:
 33:         clr.l   -(sp)
 34:         DOS     _SUPER
 35:
 36:         pea.l   abort(pc)           *中断時終了アドレス設定
 37:         move.w  #_CTRLVC,-(sp)      *
 38:         DOS     _INTVCS             *
 39:         move.w  #_ERRJVC,(sp)       *
 40:         DOS     _INTVCS             *
 41: *       addq.l  #6,sp               *
 42:
 43: .ifdef  SPEED                       *RS-232C初期化
 44:         move.w  #SPEED,d1           *
 45: .else                               *
 46:         moveq.l #-1,d1              *
 47:         IOCS    _SET232C            *
 48:         move.w  d0,d1               *
 49: .endif                              *
 50:         bsr     open232c            *
 51:         addq.w  #1,d0               *d0=-1なら初期化失敗
 52:         beq     error
```

▶アルバイトの面接の帰り，ふら〜っと日本橋へ行ったら，「現品限り大特価X68000 PRO II＋ディスプレイで送料込65,712円」なんてものを見てしまい買っちゃいました。生活があ。とにかく，これでX68000ユーザーになれて嬉しいです。　　嶋村　謙(20)大阪府

```
 53:
 54:                 lea.l   inp232c,a3      *a3 = 1文字入力ルーチン
 55:                 lea.l   buff,a4         *a4 = バッファ
 56:                 lea.l   BLOCKSIZE.w,a5  *a5 = ブロックサイズ
 57:                 moveq.l #$ff,d3         *d3 = ブロックシーケンス番号-1の補数
 58:
 59:                 move.l  a5,-(sp)        *DOS WRITE の引数
 60:                 pea.l   3-1(a4)         *
 61:                 move.w  d7,-(sp)        *
 62:
 63:                 lea.l   crctable,a2     *仮にCRC XMODEM用に
 64:                 lea.l   crc(pc),a6      *  初期設定
 65:                 move.w  #FRAMESIZE_CRC-1-1,d5
 66:
 67:                 moveq.l #4-1,d2         *CRC XMODEMが
 68: cloop:          moveq.l #'C',d1         *  利用できる?
 69:                 bsr     out232c         *
 70:                                         *
 71:                 moveq.l #3*10,d1        *
 72:                 bsr     starttimer      *
 73: cwaitloop:      bsr     inp232c         *
 74:                 cmpi.w  #SOH,d0         *
 75:                 beq     main            *
 76:                 cmpi.w  #CAN,d0         *
 77:                 beq     cancel          *
 78:                 cmpi.w  #TIMEOUTSIG,d0  *
 79:                 bne     cwaitloop       *
 80:                 dbra    d2,cloop        *
 81:
 82:                                 *チェックサムXMODENに切り替える
 83:                 lea.l   summes(pc),a1   *
 84:                 IOCS    _B_PRINT        *
 85:                 lea.l   chksum(pc),a6   *
 86:                 move.w  #FRAMESIZE-1-1,d5
 87:                 moveq.l #RETRY_C-1,d2   *
 88: nakloop:        moveq.l #NAK,d1         *
 89:                 bsr     out232c         *
 90:                                         *
 91:                 moveq.l #RETRY_T,d1     *
 92:                 bsr     starttimer      *
 93: nakwaitloop:    bsr     inp232c         *
 94:                 cmpi.w  #SOH,d0         *
 95:                 beq     main            *
 96:                 cmpi.w  #CAN,d0         *
 97:                 beq     cancel          *
 98:                 cmpi.w  #TIMEOUTSIG,d0  *
 99:                 bne     nakwaitloop     *
100:                 dbra    d2,nakloop      *
101:                 bra     abort
102: *
103: mainloop:       moveq.l #RETRY_C-1,d2   *SOHを待つ
104:                 bra     sohwait         *
105: sohloop:        moveq.l #NAK,d1         *
106:                 bsr     out232c         *
107:                                         *
108: sohwait:        moveq.l #RETRY_T,d1     *
109:                 bsr     starttimer      *
110: sohwaitloop:    bsr     inp232c         *
111:                 cmpi.w  #SOH,d0         *
112:                 beq     main            *
113:                 cmpi.w  #EOT,d0         *
114:                 beq     term            *
115:                 cmpi.w  #CAN,d0         *
116:                 beq     cancel          *
117:                 cmpi.w  #TIMEOUTSIG,d0  *
118:                 bne     sohwaitloop     *
119:                 dbra    d2,sohloop      *
120:                 bra     abort
121:
122: main:           movea.l a4,a0           *1ブロック入力する
123:                 move.w  d5,d6           *
124: inputloop:      moveq.l #1*10,d1        *
125:                 bsr     starttimer      *
126:                                         *
127:                 bsr     inp232c         *
128:                 cmpi.w  #TIMEOUTSIG,d0  *
129:                 beq     sendnak         *
130:                                         *
131:                 move.b  d0,(a0)+        *
132:                 dbra    d6,inputloop    *
133:                                         *
134:                 bsr     stoptimer       *
135:
136:                 movea.l a4,a0
137:                 move.b  (a0)+,d0        *ブロックシーケンス番号と
138:                 not.b   d0              *  その補数が
139:                 cmp.b   (a0)+,d0        *  一致することを確認する
140:                 bne     sendnak         *
141:
142:                 cmp.b   d3,d0           *前回と同じブロック?
143:                 beq     sendack         *  そうなら無視する
144:                 addq.b  #1,d0           *今回送られるべきブロック?
145:                 cmp.b   d3,d0           *
146:                 bne     abort           *  違えば中断
147:
148: chkbcc:         jsr     (a6)            *BCCを照合する
149:                 bne     sendnak         *  不一致なら再送要求
150:
151:                 DOS     _WRITE          *ファイルに書き出す
152:                 cmp.l   a5,d0           *
153:                 bne     abort           *
154:
155: next:           subq.b  #1,d3           *ブロックシーケンス番号を進める
156:                 moveq.l #ACK,d1         *次のブロックの送信を
157:                 bsr     out232c         *  要求する
158:                 moveq.l #'*',d1
159:                 IOCS    _B_PUTC
160:                 bra     mainloop
161:
162: sendack:        moveq.l #ACK,d1         *次のブロックの送信を
163:                 bsr     out232c         *  要求する
164:                 moveq.l #'-',d1
165:                 IOCS    _B_PUTC
166:                 bra     mainloop
167:
168: sendnak:        moveq.l #NAK,d1         *再送を要求する
169:                 bsr     out232c         *
170:                 moveq.l #'.',d1
171:                 IOCS    _B_PUTC
172:                 bra     mainloop
173:
174: term:           moveq.l #ACK,d1         *EOTに対するACKを送信
175:                 bsr     out232c         *
176:                 lea.l   eotmes(pc),a1
177:                 IOCS    _B_PRINT
178:                                         *ファイル末尾のEOFを削る
179:                 lea.l   2+BLOCKSIZE(a4),a0
180:                 move.w  #BLOCKSIZE-1,d6 *
181:                 moveq.l #PADCHR,d0      *
182:                 moveq.l #0,d1           *
183:                 move.w  d6,d1           *
184: eofloop:        cmp.b   -(a0),d0        *
185:                 dbne    d6,eofloop      *
186:                                         *
187:                 sub.w   d6,d1           *
188:                 neg.l   d1              *
189:                 move.l  d1,2(sp)        *
190:                 move.w  #1,6(sp)        *
191:                 DOS     _SEEK           *
192:                 clr.l   2(sp)           *
193:                 clr.l   6(sp)           *
194:                 DOS     _WRITE          *
195:                 bra     exit
196:
197: cancel:         lea.l   canmes(pc),a1   *相手の都合で中断
198:                 IOCS    _B_PRINT        *
199:                 bra     exit            *
200:
201: abort:          moveq.l #CAN,d0         *自分の都合で中断
202:                 bsr     out232c         *
203:
204: exit:           bsr     close232c
205:
206:                 moveq.l #-1,d1          *RS-232C関係IOCSリセット
207:                 IOCS    _SET232C        *
208:                 move.w  d0,d1           *
209:                 IOCS    _SET232C        *
210:
211: error:          DOS     _EXIT
212:
213: *
214: *       8ビットチェックサムを求め, 照合する
215: *
216: chksum:
217:                 moveq.l #0,d0
218:                 move.w  #BLOCKSIZE-1,d6
219: chksumloop:     add.b   (a0)+,d0
220:                 dbra    d6,chksumloop
221:
222:                 sub.b   (a0)+,d0
223:                 rts
224:
225: *
226: *       16ビットCRCを求め, 照合する
227: *
228: crc:
229:                 moveq.l #0,d0
230:                 move.w  #BLOCKSIZE+2-1,d6
231: crcloop:        ror.w   #8,d0           *つぎの8ビットを取り込む
232:                 moveq.l #0,d1           *
233:                 move.b  (a0)+,d1        *
234:                 eor.b   d0,d1           *
235:
236:                 add.w   d1,d1           *テーブルを引いて
237:                 move.w  0(a2,d1.w),d1   *  中間結果を更新する
238:                 clr.b.  d0              *
239:                 eor.w   d1,d0           *
240:
241: crcnext:        dbra    d6,crcloop      *データが尽きるまで繰り返す
242:
243:                 tst.w   d0
244:                 rts
245:
246: *
247: *       CRC計算用のテーブルを作成する
248: *
249: crc_comp:
250:                 lea.l   crctable+256*2,a0   *a0 = テーブル末尾
251:                 move.w  #CRCPOLY,d1         *d1 = 生成多項式
252:
253:                 move.w  #256-1,d2
254: comploop1:      move.w  d2,d0           *d0 = 255,254,253,...,0
255:                 lsl.w   #8,d0
256:
257:                 moveq.l #8-1,d3
258: comploop2:      add.w   d0,d0           *中間結果を1ビット左シフトする
259:                 bcc     compnext2       *第16ビットが1なら
260:                 eor.w   d1,d0           *  定数とXORを取る
261: compnext2:      dbra    d3,comploop2    *8ビット分繰り返す
262:
263:                 move.w  d0,-(a0)        *テーブルに登録する
264:
265: compnext1:      dbra    d2,comploop1    *すべてのバイト値について繰り返す
266:
267: compretn:       rts
268: *
269: eotmes:         .dc.b   '[EOT]',13,10,0
270: canmes:         .dc.b   '[CAN]',13,10,0
271: summes:         .dc.b   '[チェックサムモードへ移行]',0
272: *
273:                 .bss
274:                 .even
275: *
276: crctable:       .ds.w   256             *CRC計算用テーブル
277: buff:           .ds.b   FRAMESIZE_CRC   *入出力バッファ
278: *
279:                 .stack
280:                 .even
281: *
282:                 .ds.l   4096
283: inisp:
284:                 .end    ent
```

<対応機種一覧> ●MZ-80 K/C/700/1500 ●MZ-80 B/2000 ●MZ-2500/2861 ●X1 ●X1 turbo/Z ●PC-8001/8801/88 ●SMC-777/C ●PASOPIA/5 ●PASOPIA/7 ●FM-7/77/AV ●MSX/2/2+/turbo R ●PC-286/386/486/9801/98/9821 ●X68000/X68030
掲載されたプログラムの利用には各機種用のS-OS "SWORD" システムが必要です。

第146部 シューティングゲーム作成講座(1)

●ゲームの作り方教えます

予告どおり「YGCS ver.0.30」をベースにした「シューティングゲーム作成講座」が始まりました。

第1回目は，リアルタイムゲームの基本ともいえる，複数キャラクターの制御，移動制御を解説しています。

複数キャラクターを制御するためには，どんなワークエリアの構成をとればいいのか，YGCSの移動制御に使われているSIN関数移動とはなにか，理解することができましたでしょうか。

具体的なリストがないため，理解するのは難しいかもしれませんが，どれも基本中の基本，ゲームを作ろうとしている人は，ぜひとも覚えましょう。絶対に損はしません（というよりも，これぐらいのことを理解できないと，ゲームなんか作れないぞ！がんばって身につけるのだ）。

などといっても，理解するには多少の時間がかかるでしょう。そこで，上杉氏とその友人のご協力により，BBS上でYGCSのサポートボードが開設されることになりました(電話番号など詳しいことは112ページを参照してください)。

連載でわからなかったこと，ゲームに関して知りたいことを具体的にまとめて，どんどん質問してください。

もちろん，Oh!X編集部まで，質問をお送りいただいてもかまいません（その場合はハガキか封書でね)。Oh!X編集部が責任をもって上杉氏に渡しておきます。ただし，全部の質問に答えられるとはかぎらないので，回答がないからといって怒らないように。送られてきた質問は，どんどん誌面に反映させていくつもりです。

では，これからの「YGCS ver.0.30」と「シューティングゲーム作成講座」に期待してください。

●S-OSサークルの皆さん全員集合

ここで，いままでフリーソフト化宣言がなされたアプリケーションを，サークル活動の一環として配布してもいい，というS-OSサークルを募集します。

さらに，THE SENTINELのコーナーに積極的に参加したいサークルも募集します。力のあるサークルと判断できたときには，このTHE SENTINELのコーナーへの協力をしていただくかもしれません。

これらのサークル募集の目的は，THE SENTINELを媒体としてS-OSユーザーの輪を結びつけることにあります。

応募するにあたっては，サークル活動の概略，THE SENTINELにどのように参加したいかを適当にまとめ（文章量は自由です)，具体的に活動内容がわかる会報も一緒にお送りください。随時紹介していきたいと思います。

●今月のフリーソフト化は2本

というわけで，今月のアプリケーションのフリーソフト化にご協力していただいたものは，以下の2本です。

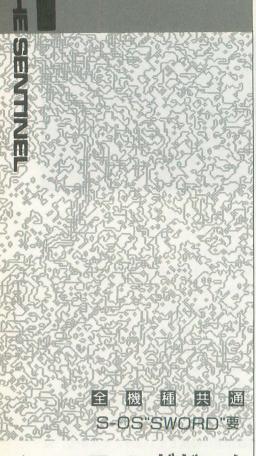
1988年9月号
第69部　超小型エディタTED-750
1991年2月号
第103部　KISMET

鈴木さん，榊さんありがとう。鈴木さんは,「いままで投稿されたもので採用されなかったものもフリーソフト化したい，けどいいのかな？」というようなことも書かれていましたが，それについてはまったく問題ありません。基本的にプログラムの著作権は作者に帰属しています。ですから，友達に配布しようが，改造しようが個人の自由です（Oh!Xに掲載されたものを改造し，配布した場合は，「XXXX年のOh!X○月号に掲載されたものを改造しました」とひと言書いておけば大丈夫)。ただし，Oh!Xに掲載された記事のコピー，配布はできません。注意してください。

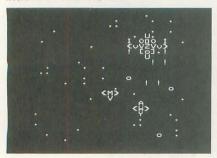
そして，「KISMET」については，「名前をそのままにして他機種へ移植する場合は連絡してください」とのこと。もしも名前をそのままにして移植を考えている人は，Oh!X編集部まで一報をお願いします。

いままでどおり,「アプリケーションのフリーソフト化計画」では，作者からの連絡を待っています。もしも，この記事が目に止まったら，ぜひともアンケートハガキでご連絡をお願いします。

1994■インデックス

全機種共通
S-OS"SWORD"要

シューティングゲーム作成講座(1)

Uesugi Yuya
上杉 悠也

今月から「YGCSver.0.30」をベースにした「シューティングゲーム作成講座」が始まります。ここでは，システムの使い方，ゲームを作るためのテクニックを解説していきます。

シューティングゲームを作る

今回から,「シューティングゲーム作成講座」を始めることになりました。といっても連載なんて初めての経験なので，不手際があるかもしれませんが，そのときは笑って許してやってください。

この講座では「YGCSver.0.30」をダシに，シューティングゲームの作り方を，基礎から解説していこうと思っています。

基本的な流れとしては，大まかな原理の説明から入って，その都度必要なプログラムを示して解説していくつもりです。

実際，ゲームを作るうえでいちばん重要なのはアルゴリズムであり，プログラムテクニックではありません。プログラムテクニックは，最低限必要なものだけ身につけておけば十分です。今回の講座ではアルゴリズムに重点をおいて話を進めていこうと思います。もちろんZ80アセンブラについての解説は特にしませんので，各自で対応してください。

で，今回はこれから必要になる基本的な概念などの説明をします。難しい話もあるとは思いますが,「ビール1本ジャンジャン持ってきてね」の意気込みで（意味不明）ついてきてください。

キャラクター管理の原点

シューティングゲームに限らず，初めてゲームを作ろうとした場合に問題になるのが，多数のキャラクターをどのようにすれば同時に動かせるかという問題だと思います（自分がそうだった)。

そこで1つのキャラを動かすところから順番に，移動に焦点を絞って解説していくことにしましょう。

●1キャラクターの場合

基本的にはなにも考えることはありません。単純に自分の動かしたいスピードで，キャラクターを動かせばいいわけです。

では，ここで出てきたスピードとはなんでしょう？　ゲーム世界でのスピードとは，1回呼び出されるたびに座標に足される値のことです。

もしこのスピードが大きすぎると，移動途中にあるキャラとの当たり判定ができません。十分注意してください。

●2キャラクターの場合

さて，待望（？）の2キャラクター同時移動です。1キャラクターの移動方法が完全に把握できていれば，2キャラクターを同時移動させるのはしごく簡単な作業です。通常1キャラクターの移動処理は図1-aのようになりますが，2キャラクターの場合は図1-bのようになります。「なんだ，当たり前のことじゃないか！」といわずに，しばらくつきあってください。

この場合，キャラクターAとキャラクターBのプログラムを続けて書いておけば，問題なく2キャラを同時移動させられます。余談ですが，私が初めて「ブロック崩し」のプログラムを作ったときも，パドルとボールの数だけプログラムを続けて書いてました（ふっ，若かったなあ)。

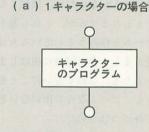
●キャラクター多数の場合

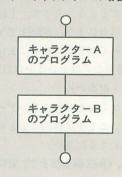
図2-aを見てください。なにやらいっぱいキャラを動かせそうですね（省略されているところに，たくさんのキャラクターのプログラムがあると思ってください)。ですが，実際はこのようにプログラムを書いても，まともにゲームを作ることはできません（と思う)。

なぜか？　それはゲーム中に出てくるキャラクターの出現数や種類が特定できないからです。もちろん特定可能であれば，ずらずらプログラムを並べても，とりあえずはゲームを作ることはできます（問題は大ありですけど)。

ではどのように作ればいいのか？

まず図2-bを見てください。このフローチャートでは，キャラクターのプログラムを続けて書くかわりに，キャラクターの種類により各プログラムへ条件分岐させています。

図1　少数キャラクターの管理

(a) 1キャラクターの場合

(b) 2キャラクターの場合

そしてこの処理部分を，出現しているキャラクター数だけループさせているわけです。aとbの違いは，aでは同時に出現しているキャラクターの数だけ（たとえ同じ種類であっても）プログラムを書かなければならなかったのに対して，bでは種類の違うキャラのプログラムだけ(ここがミソ)を書けばいいのです。もし1種類（1個ではない）のキャラしか出ない場合は，プログラムを1本しか書かなくてすむということになります。要するにこの1本で同種のキャラをすべてコントロールしてしまうわけですね。

さてさて，ここまでの話は大まかにでも理解してもらえたでしょうか？

実際にこのアルゴリズムを実現するには，ワークの構造と使い方についても知っておく必要があります。

ワークはどうする？

1本のプログラムで複数のキャラクターを制御するには，ワークを工夫する必要があります。キャラクターの座標管理を例にとって，ワークについて考えてみましょう。

通常，座標を保存しておくワークは，プログラム中にDSなどで確保しておくことを考えますが，

例)

```
        LD   A,XPOS
XPOS    DS   1；X position
YPOS    DS   1；Y position
```

これでは同時に1個のキャラクターしか管理できません。理由は次のとおりです。

いま現在，画面中に1個のキャラクターが存在していたとします。次の回に同種のキャラクターが発生（登場）しました。発生したキャラは初期値として，ワークに自分の発生座標を書き込みます。おっと！バグりました。すでに存在していたキャラクターの座標が，発生したキャラクターによって書き換えられてしまったのです。

このような事態を回避しつつ，複数のキャラを1本のプログラムで管理するには，どうしたらいいのでしょうか？

まずは図3を見てください。aがワーク全体で，bが各キャラクター用のワークです。通常，ゲームのキャラクター管理ワークというのは，このように個々のワークが集まって，全体のキャラクターワークを構成しているのです。これら個々のワークをプログラムから見て平等に扱うには，プログラムから見たアドレスを同じにすればいいわけです。

さて，もう気づいた人もいると思いますが，個々のワークの各項目（X座標とか）を参照するのに相対アドレス（個々のワークのトップが基準）を使用します。そうすることにより，どのキャラクターのワークも同じに扱うことができるようになるわけです。ただし，1つだけ注意しなければならないことがあります。個々のワークのトップアドレス（絶対アドレス）をプログラムに渡し忘れると，プログラムがどこを参照すればよいかわからなくなる点です。

実際の参照の仕方は，

例)

```
    LD A,(IX+1)；X position get
      IX＝トップアドレス
      ＋1＝相対アドレス
```

となり，キャラクターが変わった場合はトップアドレスのみを変更すればいいのです。これで1つのプログラムで，同時に複数のキャラを制御可能になります。

ここまでの概念は，機種，言語に依存しませんので，ぜひとも理解してマスターしておいてください。今後はこれらの概念をマスターしているものとして，話を進めて行きます（ちょっと不親切かな？）。

さて，移動を例に話を進めてきたので，ついでにYGCSで主に使われる移動アルゴリズムについて説明します。

SIN関数移動

これもゲーム制作においては，基本中の基本に属するテクニックといえます。

そういえば，いままでこのテクニックについて解説されたことって，ほとんどありませんね。ゲームを作るうえでかなり基本的なことなので，結構，知りたい人も多くいると思うのですが。なんででしょう？

図2　多数キャラクターの管理

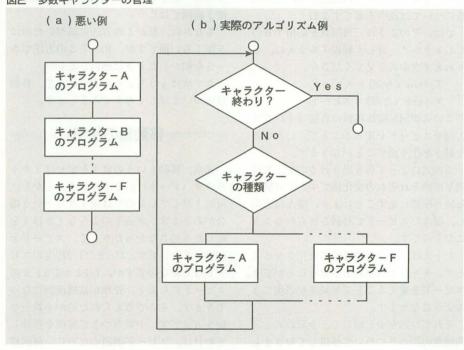
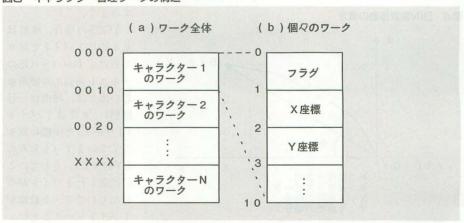



図3　キャラクター管理ワークの構造

と，余談はこれくらいにして，キャラクターの移動方法についてちょっと考えてみましょう。初めてゲームを作るときに考える方法が，単純にX座標やY座標に直接値を足していくという方法です。

しかしこの方法では基本的に直進移動しかできませんし，スピードも変えられません。かなり不便ですね。

そこで次に考えるのが，直接値を足していくかわりに変数を用いる方法です。この方法なら変数の値をいろいろ変えることにより，あらゆる動きをさせることが可能になります。もちろんスピードも自由自在に変化させられます。

さて，これでキャラクターを滑らかに動かす準備は整いました。次に必要になるのが，いかに変数の値をコントロールするかということです。うまくしないと，結局は滑らかに動かすことはできません。そこで登場してもらうのが三角関数です。三角関数については説明不要ですよね？

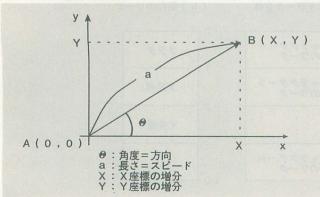
では，どのように三角関数を応用するのでしょうか？　詳しく解説するまえに，とりあえず次の式を見てください。

y＝sin（方向）×スピード
x＝cos（方向）×スピード

この式がSIN関数移動の真髄（？）です。方向やスピードを変えることで，いろいろな動きを作り出すことができます。

この式によって導き出されるのが，X座標Y座標それぞれの変化値です。この値を毎回各座標に足すことにより，望んだ方向に，望んだスピードで移動させられるようになるのです。

たとえば，方向を連続的に変化させることで，キャラに円運動をさせることができ，スピードを変えることで加減速が表現できるようになります。

それでは話を少し戻して，上記の式と三角関数の関わりについて解説しておきましょう。

では図4を見てください。これは数学のテキストでよく見かける図ですが，実はこれが重要なのです。

この図に対応した式を以下に示しておきます。

$x=\sin\theta\times a$
$y=\cos\theta\times a$

この式は線分（A，B）の角度と長さから，点Bの（X，Y）を求めるための三角関数の式です。さ，もうわかりましたね。線分（A，B）の角度が移動方向，長さが移動スピードにそれぞれ対応していて，導き出された点Bの（X，Y）がX座標とY座標の実際の移動量になるのです。

どうでしょう？　SIN関数移動の原理は理解してもらえたでしょうか？　よくわからなかった人も焦らなくて結構です。これからの講座の中で，実際の使い方を説明していきますから，そちらのほうをしっかりマスターしてもらえば十分です（YGCSを使う範囲では）。

ちなみに，私がこの方法に気づいたのは5年くらい前ですが，初めてこの方法でキャラを動かしたときは感動しました。

さて次はもうちょっと突っ込んで，移動量について詳しく考えてみましょう。

移動量の秘密

通常，移動というのは1を足せば1キャラクタ（ドット）動くものです。しかし，毎回1足していたのでは速すぎるという場合があります。普通そのようなときは1を足したり足さなかったりして，スピードを調整するわけです。が，まだ1/2程度のスピードなら楽なのですが，1/7とか2/5とかのスピードになると，管理が結構面倒になってきます。そこで考えられたのが小数点を使う方法です。小数点つきで座標を管理しておけば，スピード調節のために，毎回移動量を足すか足さないかの処理をしなくてすみます。

YGCSの場合，座標は基本的に1バイトで足りるわけですが（S-OSのテキスト画面の座標系を思い出せば，理由は一目瞭然)，実際は2バイト（1ワード）を座標に割り当てています（もちろんXY別々で）。そして，この拡張した1バイトが小数部なのです。小数部が1バイトということは，256で1キャラ（ドット）動くということになります。当然のことながら，実際の表示に使用するのは上位の1バイトのみです。

図4　SIN関数移動の概念

この小数点管理が威力を発揮するのは，前項で説明したSIN関数移動などです。三角関数演算によって導き出されたわりと細かい移動量にも，小数部があるおかげで問題なく対応できるようになります。

シューティングゲームに限らずどんなゲームでも，1と2の中間のスピードがほしいという場面はいくらでも出てくるわけですが，そんなときにこの小数点管理の方法を覚えておけば(対応しておけば)，まったく悩む必要がなくなるというわけです。

ちなみにZ80で座標を小数点管理する場合は，HLレジスタで座標計算を行い，表示の際にはHレジスタのみを使うといった感じで処理します。

以上で，「シューティングゲーム作成講座」の1回目を終わります。今回は初心者を対象に基礎的な知識を解説してきたわけですが，具体的な例を示していないのでいまいち把握しづらかったかもしれません。しかし，自分でゲームを作ろうとする場合，できるだけ知っておいてほしい知識なので，自分なりに理解できるよう努力してみてください（“システムに頼る！”という人には，その限りではありませんが）。

どうしてもわからないという人は，わかる人をつかまえて聞く，Oh!X編集部の上杉まで質問の手紙を送る，などの方法があります。が，私に直接聞きたい場合，パソコン通信で質問などを受けつけます。知り合いのシスオペさんのご好意により，後述のネットでYGCSのサポートボードを開設してもらいました。重要な質問や意見などは，できるだけ誌面に反映していこうと思っているので，気軽に立ち寄ってみてください。ちなみに，YGCSに関係ないゲーム制作に関する質問でも結構です。答えられる範囲で答えていきます。

ネット名　　：NSNW
アクセス番号：0423-90-7170
運営時間　　：24時間
備考　　　　：ゲスト書き込みOK

当然のことですが，通信を利用する場合はネットに迷惑をかけたり，番号のかけ間違いのないようにお願いします。

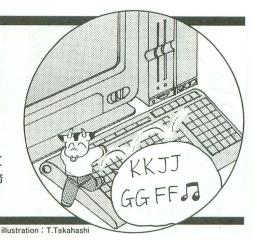
さて，次回はなにをしようかまだ考えてないのですが，サンプルを基にシステムの使い方の説明などをしていこうかと思ってます。まぁ，予定は未定であり決定ではないといいますから，楽しみに（？）待っていてください。

(で)のショートプロぱーてぃ――その58

最後のショートプロぱーてぃだ!?

Komura Satoshi 古村 聡

今月は，BASICの関数とタイピングプログラムに画面まわりのツールと盛りだくさんです。どのプログラムもリストが長めなのがちょっと残念ですね。ぷろぐらむ風まかせでは，いつもとちょっと趣向を変えて読者からの質問にお答えします。

illustration：T.Takahashi

ども～，旅から帰ってきました。いや～，旅に出ても次の締め切りまでに帰ってきて原稿書いちゃうなんて，人間ができてきたよな～，わし。いや，ほんと。

ところで久しぶりに旅に出たんですが，いつの間にか成田空港って新しくなってたんですねぇ。いつも利用してる飛行機は古いほうの第1ターミナルから出るもんで，新しい第2ターミナルには行ったことがなかったんですよ。あんなに立派になっていたんですね。第1ターミナルよりすごく広々としていて快適だったです。まだ滑走路が第1側の1本しかできてなくて，飛行機が第2ターミナルから15分くらいえんえんと地面を走っていくのはちょっと参ったけど。

それから，乗った飛行機がタイ国際航空ってのだったんですけど，これが飛行機がピッカピカで新しいんですよ。なんか，いまさっき座席のビニールカバーをはずしたような。気分よくなってしまいました。や～，新しいってええなぁ。

そういえば，瀬戸大橋のときもレインボーブリッジのときもいましたよねぇ。徹夜して一番乗りしちゃうようなおじさんとか。なんか，あーゆー人の気持ちがわかってしまった私なのであります。

……このコーナーも新しく変えちゃおうか？(おいって)

ねずみアニメだMOUSEP

ではでは今月の1本目のプログラムです。マウスカーソルを便利に使えるプログラム。MOUSEP.FNCです。どうぞっ。

MOUSEP.FNC for X680x0

(要アセンブラ，リンカ)

静岡県　伊藤直也

Human68k上だとマウスの右ボタンを1回クリック，X-BASICだとMOUSE(1)を実行すると出てくる，マウスカーソル。X-BASIC上でのマウスカーソルのパターン変更については6月号の質問箱でも取り上げていますが，このプログラムではマウスカーソルをアニメーションさせます。

このプログラムはアセンブラのソースリストの形で掲載されていますので実行するにはアセンブラ，リンカが必要になります。マクロは特にいりません(IOCS呼び出し関係も必要なものは全部リスト内に書かれていますからね)。で，リスト1をエディタで入力したあと，

A>AS MOUSEP
A>LK MOUSEP
A>REN　MOUSEP.X　MOUSEP.FNC

を実行して，BASIC.Xと同じディレクトリにプログラムをコピーします。そして，BASICの外部関数定義ファイルBASIC.CNFに，

FUNC＝MOUSEP

という行を追加してBASICからこのプログラムを呼べるようにすれば準備OKです。

このMOUSEP.FNCを使うとmsdef()とmssel()という関数が増えます。msdef()はマウスカーソルのパターンを定義します。書式は，

msdef(cd,ca,x,y)

で戻り値はありません。cdはパターンを定義するマウスカーソル番号で，4以上の値を指定してください。0～3はシステム用のため定義できません。caはパターンが格納されているchar型の配列です。プログラム上で，

int ca(256)＝{0,0,1,1,2,2,……

などとして配列を作っておいてください。0で定義した部分が透明，1と2がSRAM上に設定されている色です。Human68kだと普通はアイボリーと紺色になっていますね。x，yはマウスカーソルパターンのどこをマウスカーソルの値にするかを決めます。たとえば，(0，0)であればmsstat()などでマウスカーソルのx，y座標を調べるとマウスパターンの左上すみの部分の座標が，(8，8)であればマウスカーソルの真ん中が座標となります(図1を参照)。

もう一方のmssel()はマウスカーソルを変えるための関数です。書式は，

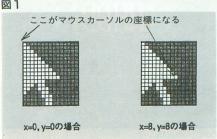
mssel([cd1]，[cd2])

です。cd1だけ指定するとマウスカーソルをそのパターンにし，mssel(13)だとマウスカーソルはパターン番号13になります。cd1,cd2の両方を指定するとcd1～cd2までのパターンでアニメーションします。たとえば，mssel(11,13)で11～13に順送りで，mssel(13,11)では逆順にアニメーションします。アニメーションパターンは最高で6つまでです。なにも指定しないとmssel(0)と同じ動作です。msdef()でパターンが定義されていないマウスカーソル番号を指定すると，番号を0にして実行されます。

む～ん，アセンブラにしては結構短い，250行あまりのプログラムでよくこれだけの機能を実現しましたね。解説(つまりこの原稿)のほうが長い気もするぞ。まぁ，よいことだ。うんうん。リスト2にはこのプログラムのサンプルが用意されています。

お金かけずにタイプ覚える!

それでは2本目のプログラムにまいりま

図1

しょう。マウスの次はキーボード。なんか入力デバイス特集みたいになってきたぞ。続いてのプログラムはX-BASIC用のゲームライクな実用的プログラム。タッチタイピングの練習をするためのTYPE_V30.BASです。どうぞっ。

TYPE_V30.BAS for X680x0

(要X-BASIC)

大分県　後藤聡文

初心者の誰もがあこがれるタッチタイピング。確かに，キーボードを見ないでタッタカ叩くあの姿。カッコいいよね。さあ，このプログラムでかっこよくキーボードを叩けるようになるのだ！

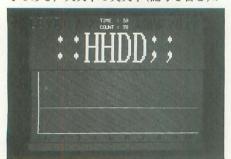
BASICを立ち上げてリスト3を入力しセーブしてください。それから，まず，データファイルを作ります。

run 5000

と入力してください。画面上に「名前を入力」と表示されます。仮に名前をDEと入力するとDE.DATというデータファイルがカレントディレクトリに作られます。2回目以降からはこのデータファイルを作る必要はありません。

さて，データファイルを作り終わったらプログラムをRUN。

名前を聞いてきますので，データファイルを作ったときの名前を入力してください。私の場合だとDEですね。それができたらいよいよゲームスタートです。おっと，その前にCAPキーをonにしてと。

レベルや目標値が画面に表示されます。そのあと，大文字の英文字(記号を含む)が

TYPE_V30.BAS

8文字表示されますので，これと同じ文字をキーボードから入力していってください。すると文字が消えてポイントが1増えます。このように1分間入力し続けます。1分たったところで目標の50文字を入れることができれば面クリア。これを2回クリアするとレベルアップします。このときにゲームを続けるかどうかを聞いてきますので続けるときにはN，やめるときにはEを入れてください。ゲームの状態は先ほどのデータファイルに記録されます。

ゲームレベルA～Bではキーボードの真ん中の列のASDFGHJKL;:の文字の中から，C～Dでは上の列のQWERTYUIOP@から，E～Fでは下の列のZXCVBNM,._から出題されます。G～Hレベルではそれまで出題されたものすべてのなかからです。

すべてをクリアしたときは感動のエンディングが待っております。

エンディングが見られたときにはきっとタッチタイピングをマスターできているはずです。難しいと思う人は140行の，

limit(2)＝{50,70,90}

の値を変えてみてください。

や～，これを入力してタッチタイピングできてテケテケとキーボードを速く叩けるようになれるといいですよね。Oh!Xの本代だけですみますもんね。私の場合，いまでもまぁ人並みのスピードではいまでも叩けると思うんですけど，ここまでのスピードになるには結構お金かかってます。

ていうのもですね，一時期チャットに凝りまくってまして。みんな叩くのが速いんですよ，キーボード。こっちも叩いて叩いて叩きまくって人並みに速くなりました。ひと月に通信代で2～3万円かかってましたけど。いまはどうなんだって？　だーいじょうぶ，自分でパソコン通信のホストを開いて，来た人をチャットに誘ってますから(う～ん，我ながらなんと凶悪な)。

画面が残るKXなのさ!

それでは今月最後のプログラムです。マウス，キーボードと来たから最後はジョイスティックだろうって？　ん～，ジョイスティックの投稿もあったんですけどね。でもこれは画面まわりの便利なプログラムです。ではどうぞっ。

KX.X for X680x0

(要アセンブラ，リンカ)

東京都　鈴木克宗

PC-9801とか一部のミニコンなどの端末のプログラムって，起動したあとも起動前の画面がちゃんと残っているんですよね。たとえば，PC-9801上でVZというエディタを起動してから終了すると，その前のDOSで操作した画面が残っているんです。X68000の場合はエディタにしても何にしても，アプリケーションを抜けたあとに前の画面が残ってることってないんですよね。でも安心。それをX68000でやってくれるプログラムがこのKX.Xなのです。

このプログラムはアセンブラのソースリストの形で掲載されています。リスト4をエディタから入力して，

A>AS KX

A>LK KX

以上のようにアセンブラ，リンカを使って実行ファイルを作ってください。

使い方は簡単で，

A>KX ［使いたいプログラム名］

です。たとえば，EDというエディタでAUTOEXEC.BATというファイルをエディットするのであれば，

A>KX ED AUTOEXEC.BAT

で，エディットしたあとで画面を元に戻すことができます。

エイリアス機能(「Human68kユーザーズマニュアル」参照)で，

A>ED KX ED［ESC］［I］

などとしておけば次からは，

A>ED

と入力するだけで，常にKXを使ってEDを起動できます。ただし，このKXでは文字の色はすべて白で表示されます。

なぁるほど，圧縮なんですね。

この手のプログラムって，要するにアプリケーションを起動する前に画面の内容を保存しておいて，アプリケーションの終了後にその内容を表示するだけなんですけど，問題はX68000のテキストVRAMの容量なのです。X68000の場合，全部で512Kバイト(PC-9801の場合，一度の表示に使われているのは32Kバイト)もあるので待避するのが非常に大変なんです。512Kバイトもあるとメインメモリに待避する場合，ほかのアプリケーションが動かなくなる可能性もあります。それに，ディスクだとスペースはあるけど時間がかかりすぎてうっとうしいですし。実をいうと私もこういうプログラムを作ろうとしたんですけど，そこでつまっちゃったんですよね。

このプログラムではデータを圧縮することでメインメモリに待避できるようにしてあるんですね。

圧縮といっても処理に時間をかけるわけにはいかないので，そのアルゴリズムは割

と簡単で，各ラスタについて先頭から0でない最後のワードまでを保存しているだけです。テキストは通常左から順に文字が書かれていきますし，たいていは画面の半分も使われませんから，これだけの作業をするだけでもかなり効果があるんですね。なにも表示されていないラスターはほとんどメモリを消費しないので，特にドット数の多い画面モードで使っている場合ほど圧縮率が高いというのも好都合ですし。

で，ほとんどの文字は白で表示されているので，このプログラムでは色つき文字を無視することにしています。つまり，テキストプレーン0のみを保存し，復元するときにはテキストプレーン0と1に同時に書き込んでいるのです。これでデータ量と処理時間が半分になりますからね。文字が全部白になるのはそういうことなのですね。

さーて，古いショートプロぱーてぃはこれでおしまい。では新しいショートプロぱーてぃで！　　(いいのか本当に(苦笑))

リスト1　MOUSEP.S

```
  1: *============================================*
  2: *  マウスパターン設定関数 (mousep.fnc)
  3: *============================================*
  4: _MS_PATST  equ     $7a     * IOCS ｺｰﾙ
  5: _MS_SEL            equ     $7b
  6: _MS_SEL2   equ     $7c
  7:
  8: char_val   equ     $0004   * ﾃﾞｰﾀ型1
  9: char_ary1  equ     $0034
 10: char_omt   equ     $0084
 11: void_ret   equu    $ffff
 12:
 13: omit               equ     -1      * 省略
 14:
 15: *============================================*
 16: *  ジャンプテーブル等
 17: *============================================*
 18:    .even
 19:    .dc.l   e_dmy,e_dmy     * ﾀﾞﾐｰ * 8
 20:    .dc.l   e_dmy,e_dmy
 21:    .dc.l   e_dmy,e_dmy
 22:    .dc.l   e_dmy,e_dmy
 23:    .dc.l   t_fncnm         * 関数名 ﾃｰﾌﾞﾙ
 24:    .dc.l   t_param         * ﾊﾟﾗﾒｰﾀ ﾃｰﾌﾞﾙ
 25:    .dc.l   t_exec          * 実行ｱﾄﾞﾚｽ ﾃｰﾌﾞﾙ
 26:    .dc.l   0,0,0,0,0
 27:
 28: e_dmy:     rts                     * 何もしない
 29:
 30: *============================================*
 31: *  関数名
 32: *============================================*
 33:    .even
 34: t_fncnm:
 35:    .dc.b   'mssel',0
 36:    .dc.b   'msdef',0
 37:    .dc.b   0
 38:
 39: *============================================*
 40: *  各関数のパラメータの型
 41: *============================================*
 42:    .even
 43: t_param:
 44:    .dc.l   p_sel
 45:    .dc.l   p_def
 46:
 47: p_sel:
 48:    .dc.w   char_omt        * in :省略可能
 49:    .dc.w   char_omt        * in :省略可能
 50:    .dc.w   void_ret        * out:なし
 51: p_def:
 52:    .dc.w   char_val                * in :必須
 53:    .dc.w   char_ary1       * in :必須
 54:    .dc.w   char_omt        * in :省略可能
 55:    .dc.w   char_omt        * in :省略可能
 56:    .dc.w   void_ret        * out:なし
 57:
 58: *============================================*
 59: *  各関数の実行アドレス
 60: *============================================*
 61:    .even
 62: t_exec:
 63:    .dc.l   e_sel
 64:    .dc.l   e_def
 65:
 66: *============================================*
 67: *  mssel()
 68: *============================================*
 69: e_sel:
 70:    moveq.l #0,d5           * ﾊﾟﾗﾒｰﾀ下限
 71:    moveq.l #15,d6          * ﾊﾟﾗﾒｰﾀ上限
 72:    clr.l   d1              * 後でﾊﾟﾗﾒｰﾀが入る
 73:
 74:    cmpi.w  #omit,16(sp)
 75:    bne     e_sel2          * ﾊﾟﾗﾒｰﾀ2個?
 76:    cmpi.w  #omit,6(sp)
 77:    beq     e_sel1          * ﾊﾟﾗﾒｰﾀ1個?
 78:
 79: *----      パラメータ1個の処理    -------------*
 80:    move.l  12(sp),d1       * 第一パラメータ
 81:    bsr     s_limit         * 範囲チェック
 82:    tst.w   d0
 83:    bne     e_sel_err1      * 範囲エラー?
 84: e_sel1:
 85:    moveq.l #_MS_SEL,d0
 86:    trap    #15             * ﾊﾟﾀｰﾝ ｾﾚｸﾄ
 87:    clr.l   d0              * 正常終了
 88:    rts
 89:
 90: *----      パラメータ2個の処理    -------------*
 91: e_sel2:
 92:    cmpi.w  #omit,6(sp)
 93:    beq     e_sel3          * 第一ﾊﾟﾗﾒｰﾀ省略?
 94:    move.l  12(sp),d1       * 第一ﾊﾟﾗﾒｰﾀ
 95:    bsr     s_limit         * 範囲チェック
 96:    bne     e_sel_err1      * 範囲エラー?
 97: e_sel3:
 98:    move.w  d1,d2           * d2=第一ﾊﾟﾗﾒｰﾀ
 99:
100:    move.l  22(sp),d1       * 第二ﾊﾟﾗﾒｰﾀ
101:    bsr     s_limit         * 範囲チェック
102:    bne     e_sel_err1      * 範囲エラー?
103:    move.w  d1,d3           * d3=第二ﾊﾟﾗﾒｰﾀ
104:
105:    lea.l   anim,a0         * ﾊﾟﾀｰﾝﾅﾝﾊﾞｰﾃｰﾌﾞﾙ
106:    moveq.l #1,d1           * 増分値
107:    move.w  d3,d4
108:    sub.w   d2,d4           * d3-d2=d4 (ｶｳﾝﾀ)
109:    bge     e_sel4          * 逆再生?
110:    moveq.l #-1,d1          * 増分値
111:    neg.w   d4              * カウンタ反転
112:
113: e_sel4:
114:    cmpi.w  #6,d4
115:    bge     e_sel_err2      * 6ｺﾏを越えいる
116:
117: e_sel5:
118:    move.w  d2,(a0)+        * ﾊﾟﾀｰﾝﾅﾝﾊﾞｰ転送
119:    add.w   d1,d2           * ﾊﾟﾀｰﾝﾅﾝﾊﾞｰ ｶｳﾝﾄ
120:    dbra    d4,e_sel5       * 転送数 ｶｳﾝﾄ
121:    move.w  #-1,(a0)        * ｽﾄｯﾊﾟｰ
122:
123:    lea.l   anim,a1         * ﾊﾟﾀｰﾝﾅﾝﾊﾞｰ ｱﾄﾞﾚｽ
124:    moveq.l #_MS_SEL2,d0
125:    trap    #15             * 再生
126:
127:    clr.l   d0              * 正常終了
128:    rts
129:
130: *----      エラー処理      ---------------------*
131: e_sel_err1:
132:    moveq.l #1,d0
133:    lea.l   msg1,a1
134:    rts
135:
136: e_sel_err2:
137:    moveq.l #1,d0
138:    lea.l   msg3,a1
139:    rts
140:
141: *============================================*
142: *  msdef()
143: *============================================*
144: *----      ﾊﾟﾀｰﾝ番号より格納ｱﾄﾞﾚｽを求める  -----*
145: e_def:
146:    moveq.l #4,d5           * ﾊﾟﾗﾒｰﾀ下限
147:    moveq.l #15,d6          * ﾊﾟﾗﾒｰﾀ上限
148:    move.l  12(sp),d1       * ﾊﾟﾀｰﾝﾅﾝﾊﾞｰ
149:    bsr     s_limit         * 範囲チェック
150:    bne     e_def_err2      * 範囲エラー?
151:
152:    move.l  #pattbl,d0      * ﾊﾟﾀｰﾝﾃｰﾌﾞﾙ
153:    subq.w  #4,d1           * ﾊﾟﾀｰﾝﾅﾝﾊﾞｰ -4
154:    muls.w  #68,d1          * ｵﾌｾｯﾄ計算
155:    add.l   d1,d0           * 格納ｱﾄﾞﾚｽ計算
156:    movea.l d0,a0           * 格納ｱﾄﾞﾚｽ
157:    movea.l d0,a1           * 格納ｱﾄﾞﾚｽ保存
158:
159: *----      ｷｬﾗｸｸ内のﾏｳｽｶｰｿﾙの座標  -------------*
160:    moveq.l #0,d5           * ﾊﾟﾗﾒｰﾀ下限
161:    moveq.l #15,d6          * ﾊﾟﾗﾒｰﾀ上限
162:
163:    clr.l   d1              * 後でﾊﾟﾗﾒｰﾀが入る
164:    cmpi.l  #omit,26(sp)
165:    beq     e_def1          * 第三ﾊﾟﾗﾒｰﾀ省略?
166:    move.l  32(sp),d1       * 第三ﾊﾟﾗﾒｰﾀ
167:    bsr     s_limit         * 範囲チェック
168:    bne     e_def_err1      * 範囲エラー?
169: e_def1:
170:    move.w  d1,(a0)+        * ﾏｳｽｶｰｿﾙ x
171:
172:    clr.l   d1              * 後でﾊﾟﾗﾒｰﾀが入る
173:    cmpi.l  #omit,36(sp)
174:    beq     e_def2          * 第四ﾊﾟﾗﾒｰﾀ省略?
175:    move.l  42(sp),d1       * 第四ﾊﾟﾗﾒｰﾀ
176:    bsr     s_limit         * 範囲チェック
177:    bne     e_def_err1      * 範囲エラー?
178: e_def2:
179:    move.w  d1,(a0)+        * ﾏｳｽｶｰｿﾙ y
180:
181: *----      データ変換      ---------------------*
182:    movea.l a0,a2           * a0=ﾃﾞｰﾀｱﾄﾞﾚｽ1
183:    add.l   #32,a2          * a1=ﾃﾞｰﾀｱﾄﾞﾚｽ2
184:    movea.l 22(sp),a3
185:    add.l   #10,a3          * 入力ﾃﾞｰﾀｱﾄﾞﾚｽ
186:    moveq.l #15,d5          * ｶｳﾝﾄ1
187:
188: e_def_lp1:
```

```
189:    moveq.l #15,d6              * カウント2
190:    clr.w   d0                  * 組立用データクリア
191:    clr.w   d2
192:
193: e_def_lp2:
194:    lsl.w   #1,d0               * 最初にずらす
195:    lsl.w   #1,d2
196:    move.b  (a3)+,d3            * 入力データ読む
197:
198:    cmpi.b  #1,d3
199:    beq     e_def_sk            * d3=1 ?
200:    addq.w  #1,d0               * プレーン1
201:
202:    tst.b   d3
203:    beq     e_def_sk            * d3=0 ?
204:    addq.w  #1,d2               * プレーン2
205:
206: e_def_sk:
207:    dbra    d6,e_def_lp2        * カウント1
208:
209:    move.w  d0,(a0)+            * データ セット
210:    move.w  d2,(a2)+            * データ セット
211:    dbra    d5,e_def_lp1        * カウント2
212:
213: *----          パターン定義          ----------------------*
214:    move.l  12(sp),d1           * パターンナンバー
215:    moveq.l #_MS_PATST,d0
216:    trap    #15                 * パターンセット
217:    clr.l   d0                  * 正常終了
218:    rts
219:
220: *----          エラー処理          ----------------------*
221: e_def_err1:
222:    moveq.l #1,d0
223:    lea.l   msg1,a1
224:    rts
225:
226: e_def_err2:
227:    moveq.l #1,d0
228:    lea.l   msg2,a1
229:    rts
230:
231: *==========================================*
232: *   範囲チェック サブルーチン
233: *==========================================*
234: s_limit:
235:    cmp.l   d1,d5
236:    bgt     s_limit_err         * d1<d5 (下限)
237:    cmp.l   d1,d6
238:    blt     s_limit_err         * d1>d6 (上限)
239:
240:    clr.w   d0                  * エラーフラグ off
241:    rts
242:
243: s_limit_err:
244:    moveq.l #1,d1               * エラーフラグ on
245:    rts
246: *==========================================*
247: *   ワークエリア
248: *==========================================*
249:    .even
250: *----          エラーメッセージ
251: msg1:          .dc.b   'パラメータの値が規定外です。',0,0
252: msg2:          .dc.b   'パターン番号が4~15以外です。',0,0
253: msg3:          .dc.b   '再生コマ数が6を越えています。',0,0
254:
255: *----          アニメパターンナンバー
256: anim:          .dc.w   1,2,3,4,5,6,-1
257:
258: *----          パターンバッファ          (2+2+32+32)*12
259: pattbl:        .ds.b   816
260:
261:    .end
```

リスト2 SAMPLE.BAS

```
 10 /*  mousep.fnc サンプルプログラム
 20 /*
 30 int i,x,y,l,r
 40 dim char a(255)  /* char型で255を指定
 50 screen 1,0,1,1: color 3:cls
 60 mouse(0):mouse(1):setmspos(256,256)
 70 pat()
 80 /*
 90 while 1
100   mssel(4)            /* パターン4を選択
110   locate 20,0:print"左クリックしてごらん"
120   l=0
130   while l=0
140     msstat(x,y,l,r)
150   endwhile
160   mssel(5,8)          /* 5~8を再生
170   locate 20,0:print"ほ~らアニメだよ!?"
180   for i=0 to 50000:next
190 endwhile
200 end
210 /*****************************************/
220 func pat()
230   a={ 0,2,2,0,0,0,0,0,0,0,2,2,0,0,0,
240       2,1,1,2,0,0,0,0,0,0,2,1,1,2,0,0,
250       2,1,1,1,2,0,0,0,0,2,1,1,1,2,0,0,
260       0,2,1,1,1,2,0,0,0,2,1,1,2,0,0,0,
270       0,0,2,1,1,1,2,0,2,1,1,1,2,0,0,0,
280       0,0,0,2,1,1,1,2,1,1,1,1,2,0,0,0,
290       0,0,0,0,2,1,1,1,1,1,1,1,2,0,0,0,
300       0,0,0,2,1,2,1,1,1,1,1,1,1,2,0,0,
310       0,0,0,2,1,1,2,1,1,1,1,1,1,2,0,0,
320       0,0,2,1,2,1,1,1,1,1,1,1,1,2,0,0,
330       0,0,2,1,1,2,1,1,1,1,1,1,1,2,0,0,
340       0,0,2,2,1,1,1,1,1,1,1,1,2,0,0,0,
350       0,0,2,1,2,1,1,1,1,1,1,1,2,0,0,0,
360       0,0,0,2,1,1,1,1,1,1,1,2,0,0,0,0,
370       0,0,0,0,2,1,1,1,1,1,2,0,0,0,0,0,
380       0,0,0,0,0,2,2,2,2,2,0,0,0,0,0,0 }
390   msdef(4,a)          /* マウスパターン定義
400   /*
410   a={ 0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,
420       0,1,2,1,0,0,0,0,0,0,0,0,1,2,1,0,
430       1,2,2,1,1,0,0,0,0,0,0,1,1,2,2,1,
440       0,1,1,1,2,1,0,0,0,0,1,2,1,1,1,0,
450       0,0,1,2,2,1,1,0,0,1,1,2,2,1,0,0,
460       0,0,1,1,1,1,2,1,1,2,1,1,1,1,0,0,
470       0,1,2,2,2,1,2,2,2,2,1,2,2,2,1,0,
480       1,2,1,1,1,2,1,2,2,1,2,2,2,2,2,1,
490       1,2,1,1,1,2,1,2,2,1,2,2,2,2,2,1,
500       0,1,2,2,2,1,2,2,2,2,1,2,2,2,1,0,
510       0,0,1,1,1,1,2,1,1,2,1,1,1,1,0,0,
520       0,0,1,2,2,1,1,2,2,1,1,2,2,1,0,0,
530       0,1,1,1,2,1,1,2,2,1,1,2,1,1,1,0,
540       1,2,2,1,1,0,1,2,2,1,0,1,1,2,2,1,
550       0,1,2,1,0,0,1,2,2,1,0,0,1,2,1,0,
560       0,0,1,0,0,0,1,2,2,1,0,0,0,1,0,0 }
570   msdef(5,a,8,8): msdef(6,a,8,8)
580   /*
590   a={ 0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,
600       0,1,2,1,0,0,0,0,0,0,0,0,1,2,1,0,
610       1,2,2,1,1,0,0,0,0,0,0,1,1,2,2,1,
620       0,1,1,1,2,1,0,0,0,0,1,2,1,1,1,0,
630       0,0,1,2,2,1,1,0,0,1,1,2,2,1,0,0,
640       0,0,1,1,1,1,2,1,1,2,1,1,1,1,0,0,
650       0,1,2,2,2,1,2,2,2,2,1,2,2,2,1,0,
660       1,2,2,2,2,2,1,2,2,1,2,1,1,1,2,1,
670       1,2,2,2,2,2,1,2,2,1,2,1,1,1,2,1,
680       0,1,2,2,2,1,2,2,2,2,1,2,2,2,1,0,
690       0,0,1,1,1,1,2,1,1,2,1,1,1,1,0,0,
700       0,0,1,2,2,1,1,2,2,1,1,2,2,1,0,0,
710       0,1,1,1,2,1,1,2,2,1,1,2,1,1,1,0,
720       1,2,2,1,1,0,1,2,2,1,0,1,1,2,2,1,
730       0,1,2,1,0,0,1,2,2,1,0,0,1,2,1,0,
740       0,0,1,0,0,0,1,2,2,1,0,0,0,1,0,0 }
750   msdef(7,a,8,8): msdef(8,a,8,8)
760 endfunc
```

リスト3 TYPE_V30.BAS

```
 10 /*****************************************************/
 20 /*  TYPE.BAS   Ver.3.0                   */
 30 /*  FOR X68k   BY 後藤 聡文              */
 40 /*****************************************************/
 50 int i,j,k,m,n,x,y,rr,temp,end_flag,lvt
 60 dim int lvp(2)={ 0,11,22 },c_data(2)={15,7,13}
 70 int gmiss(11)={1,0,0,0,0,0,0,0,0,0,0,0}
 80 dim str moji(32)[1]={ "A","S","D","F","G","H","J","K","L",
";",":","Q","W","E","R","T","Y","U","I","O","P","@","Z","X","C",
"V","B","N","M",",",".","/","_" }
 90 str na
100 dim str ss(8),lv2(2)={"1","2","3"}
110 dim str lv4(9)={"A","B","C","D","E","F","G","H"}
120 dim str gs2(9)={ " 1"," 2"," 3"," 4"," 5"," 6"," 7"," 8","
9","10" }
130 dim int lv(1),lvk(1),g_time(1),count(1)
140 dim int limit(2)={ 50,70,90 }
150 dim str lv3(2)={"50","70","90"}
160 screen 1,1,1,1:console ,,0
170 music()
180 graph1()
190 rnd_rw(1)
200 graph3()
210 repeat
220         lvt=lvk(0)
230         game()
240         rnd_rw(0)
250         end_flag=text()
260         if lvt<>lvk(0) then wipe()
270 until end_flag=1
280 /*
290 if lv(0)=-1 then ending() else width 64
300 end
310 /*
320 /*     サブルーチン
330 /*
340 func game()
350       m_write()
360       if lvk(0)=3 then {
370          lvk(0)=0
380          lv(0)=lv(0)+1
390          if lv(0)=8 then lv(0)=-1
400       }
410 endfunc
420 /*
430 func rnd_rw(w)
440       int bi
450       dim int h(1),temp(1)
```


▶大学の実験のため，X68000より某国民機の使用時間のほうが長い。誰かが「ぷよぷよ」を対戦してくれればX68000も稼働するのに……。　河合　利憲(21)岐阜県

```
 460        if w=1 then { color 2
 470           print "     TYPE.BAS VER.3.0"
 480           color 3:input "    名前を入力 ",na
 490           bi=fopen(na+".dat","r")
 500           cls
 510           for i=0 to 10
 520              fread(temp,1,bi)
 530              gmiss(i)=temp(0)
 540           next
 550           fread(temp,1,bi):lv(0)=temp(0):fread(temp,1,bi):lv
k(0)=temp(0):fread(temp,1,bi):lvk(1)=temp(0)
 560        } else {
 570           bi=fopen(na+".dat","w")
 580           for i=0 to 10
 590              temp(0)=gmiss(i)
 600              fwrite(temp,1,bi)
 610           next:temp(0)=lv(0)
 620           fwrite(temp,1,bi):temp(0)=lvk(0):fwrite(temp,1,bi)
:temp(0)=lvk(1):fwrite(temp,1,bi) }
 630           fclose(bi)
 640  endfunc
 650  /*
 660  func int compare()
 670        int i,j,mm
 680        str kb,as
 690        int flag,ti,ttime
 700        i=0
 710  /* リアルタイムキー入力 : キーバッファクリア
 720        while(keysns()):kb=inkey$(0):endwhile
 730  /*
 740        symbol(205,10,"TIME  :",1,1,1,13,0)
 750        symbol(205,35,"COUNT :",1,1,1,15,0)
 760        while i<8
 770           flag=0
 780           repeat
 790              if keysns() then as=inkey$:flag=1
 800              time_ch(1)
 810              ti=g_time(1)-g_time(0)
 820              if ti-ttime>=1 then {
 830                 fill(265,5,290,30,0)
 840                 symbol(270,10,itoa(ti),1,1,1,13,0)
 850                 ttime=ti
 860              }
 870              if g_time(1)-g_time(0)>=60 then return (-1)
 880           until flag=1
 890           if ss(i)="_" then {
 900              if as=" " then {
 910                 symbol(70+i*45,45,ss(i),4,6,2,0,0)
 920                 i=i+1:count(0)=count(0)+1:continue
 930           } }
 940           if ss(i)=as then {
 950              symbol(70+i*45,45,ss(i),4,6,2,0,0)
 960              i=i+1:count(0)=count(0)+1
 970           } else {
 980              beep:count(0)=count(0)-1
 990           }
1000           fill(265,30,295,55,0)
1010           symbol(270,35,itoa(count(0)),1,1,1,15,0)
1020        endwhile
1030        return(0)
1040  endfunc
1050  /*
1060  func graph1()
1070        apage(3)
1080        fill(0,0,511,511,8)
1090        for m=0 to 4
1100           box(1+m,196+m,510-m,480-m,4)
1110        next
1120        for m=0 to 1
1130           line(30+m,205  ,30+m,440  ,9)
1140           line(30  ,440+m,490 ,440+m,9)
1150        next
1160        line(32,340,490,340,5)
1170        for m=0 to 9
1180           symbol(35+m*46,445,gs2(m),1,1,2,2,0)
1190        next
1200  endfunc
1210  /*
1220  func graph3()
1230        float hi,lm
1240        apage(2):wipe()
1250        lm=limit(lvk(0))
1260        tx=53:hi=gmiss(1)/lm:ty=440-hi*100
1270        for i=1 to gmiss(0)-1
1280           lm=limit(lvk(0))
1290           count(0)=gmiss(i):hi=count(0)/lm
1300           circle( 7+i*46,440-hi*100,3,c_data(lvk(0)))
1310           line(tx,ty, 7+i*46,440-hi*100,c_data(lvk(0)))
1320           tx= 7+i*46:ty=440-hi*100
1330        next
1340  endfunc
1350  /*
1360  func graph2()
1370        float hi
1380        int tx,ty
1390        gmiss(gmiss(0))=count(0):gmiss(0)=gmiss(0)+1
1400        if gmiss(0)>11 then {
1410           for i=1 to 10
1420              gmiss(i)=gmiss(i+1)
1430           next:gmiss(0)=11
1440        }
1450  endfunc
1460  /*
1470  func int text()
1480        str ne,kb
1490        while (keysns()):kb=inkey$(0):endwhile
1500        if lv(0)=-1 then return (1)
1510        repeat
1520              cls:locate 3,5
1530              input "NEXT [N] : END [E]",ne
1540        until ne="N" or ne="n" or ne="E" or ne="e"
1550        cls
1560        if ne="E" or ne="e" then return(1)
1570        if ne="N" or ne="n" then return(0)
1580  endfunc
1590  /*
1600  func time_ch(jj)
1610        str as,h,m,s
1620        int h2,m2,s2
1630        as=time$
1640        h=mid$(as,1,2)
1650        m=mid$(as,4,2)
1660        s=mid$(as,7,2)
1670        h2=atoi(h)
1680        m2=atoi(m)
1690        s2=atoi(s)
1700        g_time(jj)=h2*3600+m2*60+s2
1710  endfunc
1720  /*
1730  func m_fill()
1740        apage(1):wipe()
1750        symbol(0,5,"LEVEL "+lv4(lv(0))+"-"+lv2(lvk(0)),1,2,
2,2,0)
1760        symbol(300,5,"1分間に"+lv3(lvk(0))+"文字",1,2,2,2,0)
1770  endfunc
1780  /*
1790  func check_level()
1800        if limit(lvk(0))<=count(0) then { music_play1()
1810           if lvk(1)=0 then { lvk(1)=1
1820           } else { lvk(1)=0:lvk(0)=lvk(0)+1:gmiss(0)=1:for i
=1 to 10:gmiss(i)=0:next:music_play3() } } else { lvk(1)=0:music
_play2() }
1830  endfunc
1840  /*
1850  func m_write()
1860        int t
1870        m_fill()
1880        locate 25,7:print "Hit any key."
1890        repeat:until keysns()
1900        cls
1910        time_ch(0)
1920        count(0)=0:apage(0)
1930        repeat
1940           if (lv(0) mod 2)<>0 then {
1950              for i=0 to 7
1960                 rr=mj_rr()
1970                 ss(i)=moji(rr)
1980                 symbol(70+i*45,45,ss(i),4,6,2,14,0)
1990              next
2000           } else {
2010              i=0:j=0
2020              while j<8
2030                 rr=mj_rr()
2040                 j=j+2
2050                 ss(i)=moji(rr):ss(i+1)=moji(rr)
2060                 symbol(70+i*45,45,ss(i),4,6,2,14,0):i=i+1
2070                 symbol(70+i*45,45,ss(i),4,6,2,14,0):i=i+1
2080              endwhile
2090           }
2100           t=compare()
2110        until t=-1
2120        wipe():graph2()
2130        graph3():check_level()
2140  endfunc
2150  /*
2160  func mj_rr()
2170        int rr
2180        if lv(0)<=5 then {
2190        rr=rnd()*11+lvp(lv(0)/2)
2200        } else {
2210        rr=rnd()*33
2220        }
2230        return(rr)
2240  endfunc
2250  /*
2260  func music()
2270        m_init()
2280        m_alloc(1,1000):m_alloc(2,1000):m_alloc(3,1000)
2290        m_assign(1,1):m_assign(2,2):m_assign(3,3)
2300        m_trk(1,"@1o5v15 g16f16a4")
2310        m_trk(2,"@1o3v15 f8e8d8c2")
2320        m_trk(3,"@1o5v15 g16f16a4g16a16b4")
2330  endfunc
2340  /*
2350  func music_play1()
2360        m_play(1)
2370  endfunc
2380  /*
2390  func music_play2()
2400        m_play(2)
2410  endfunc
2420  /*
2430  func music_play3()
2440        m_play(3)
2450  endfunc
2460  /*
2470  func ending()
2480        apage(0):cls
2490        fill(0,50,511,190,6)
2500        symbol(0,35,"CONGRATULATION",3,7,2,13,0)
2510  endfunc
2520  /*
2530  /*  このプログラムを実行する前に、" run 5000 "と
2540  /*  打ち込んでリターンキーを押して下さい 。
2550  /*  それから、"名前を入力" と表示されますので、
2560  /*  名前を入力してリターンキーを押して下さい。
2570  /*  "<名前>.dat" というファイルが作られます。
2580  /*  ************************
2590  /*  "run" と入力してリターンキーを押して下さい。
2600  /*  "名前を入力" と表示されたら、さっき入力した
2610  /*  名前を入力して下さい。
2620  /*  ゲームでの成績は、"<名前>.dat" に記録され
```

▶この不況，早くどないかして。　　　　北浦　真一(24)大阪府

```
2630 /*  ます。
2640 /*  ゲームを始めるのが2回目以降のときは、***の
2650 /*  所から始めて下さい。
5000       str ms,kb,na
5010       int ai,i
5020       dim int p(1),temp(1)
5030       while(keysns()):kb=inkey$(0):endwhile
5040       width 64:color 2
5050       print "          TYPE.BAS VER.3.0"
5060       color 3:input "          名前を入力 ",na
5070       print "",na,".dat を新規ファイルとしてオープンします"
5080       ai=fopen(na+".dat","c")
5090       temp(0)=1:fwrite(temp,1,ai)
5100       temp(0)=0
5110       for i=1 to 10
5120          fwrite(temp,1,ai)
5130       next
5140       temp(0)=0:fwrite(temp,1,ai):fwrite(temp,1,ai):fwrite(
temp,1,ai)
5150       fwrites(na,ai)
5160       fclose(ai)
5170 end
```

リスト4 KX.S

```
  1: * kx.s        Keep screen image & eXecute command
  2: *          テキスト画面を保存後、指定のコマンドを起動
  3: *          コマンド実行後に画面を復帰する。Human68k ver2 以降専用
  4: * v0.10 Nov 3  1993 suzu-kat
  5:
  6: * 終了コード
  7: NOPARERR      equ      $0101 * 実行ファイルが指定されなかった時
  8: OSVERERR      equ      $0102 * Human68k のバージョンが 2.00 以上でない
時
  9: FNCALLER      equ      $0300 * ファンクションコール時にエラー発生。
 10: *                             (※下位 8bits はファンクションコールエラー
コード)
 11: *                      $xxxx    子プロセスを起動した時はその終了コード
 12:
 13:   .include doscall.mac
 14:   .include iocscall.mac
 15:
 16: TVRAM         equ      $E00000 * テキスト VRAM
 17: SAVLINES      equ      1024    * セーブするライン数
 18: NEXTLINE      equ      128     * 1ラインのサイズ
 19: WORKSIZ       equ      SAVLINES*129 * 消費しうる最大のバッファサイズ
 20: CRTC_R21      equ      $e8002a
 21:
 22: DOS_S_MALLOC macro
 23:   .local ?v2,?1
 24:       cmp.w    #$0300,sysver
 25:       bcs      ?v2
 26:       dc.w     $ffad * ver3.00 以上
 27:       bra      ?1
 28: ?v2:
 29:       dc.w     $ff7d * ver2.xx
 30: ?1:
 31:   endm
 32:
 33: PUTS macro str
 34:   .local ?1
 35:       pea.l    ?1
 36:       DOS      _PRINT
 37:       addq.l   #4,sp
 38:   .data
 39: ?1: dc.b str,13,10,0
 40:   .even
 41:   .text
 42:   endm
 43:
 44: CHKARG macro * パラメータのチェック
 45:   .local ?1
 46:       tst.b    (a2)
 47:       bne      ?1
 48:       PUTS     "使用法: kx 実行ファイル [パラメータ..]"
 49:       PUTS     "機能: 画面を保存しつつ、指定のファイルを実行する."
 50:       move.w   #NOPARERR,-(sp)
 51:       DOS      _EXIT2
 52: ?1:
 53:   endm
 54:
 55: ERREXIT macro code,mes * ファンクションコールのエラーのため終了
 56:   .local ?1,?2
 57:       and.w    #$0ff,code
 58:       or.w     #FNCALLER,code
 59:       PUTS     mes
 60:       move.w   code,-(sp)
 61:       DOS      _EXIT2
 62:   endm
 63:
 64:   .text
 65:   .even
 66: errver:
 67:       PUTS     "このプログラムは Human68k ver2.00 以降専用です."
 68:       move.w   #OSVERERR,-(sp)
 69:       DOS      _EXIT2
 70:
 71: errmem:
 72:       ERREXIT d0,"kx: ワークエリアが確保できません."
 73:
 74: errmem2:
 75:       move.l   d0,d1
 76:       move.l   workptr,-(sp)
 77:       DOS      _MFREE
 78:       addq.l   #4,sp
 79:       ERREXIT d1,"kx: 画面保存エリアが確保できません."
 80:
 81: errexec0:
 82:       ERREXIT d0,"kx: コマンドの実行ができません."
 83:
 84: errexec2:
 85:       ERREXIT d0,"kx: 指定のファイルが見つかりません."
 86:
 87: countdata:   * 保存するデータ長を調べる
 88: * in:   a0.l 調べるラインのアドレス
 89: * out:  d0.l 保存する必要のある長さ(words)
 90: reg = a0
 91:       move.l   reg,-(sp)
 92:       lea.l    NEXTLINE(a0),a0
 93:       move.l   #1024/16,d0      * initial data length
 94: _ceL1:
 95:          tst.l    -(a0)
 96:          bne      _ceL2
 97:          subq.l   #2,d0
 98:       bhi _ceL1
 99: _ceL2:
100:       movea.l  (sp)+,reg
101:       rts
102:
103: savetvram:  * Text VRAM を (workptr)~ に待避する
104: reg = d1/d5/a0/a4
105:       movem.l reg,-(sp)
106:       move.l   datasiz,d5
107:       clr.l    -(sp)
108:       DOS      _SUPER
109:       move.l   d0,(sp)
110:
111:       movea.l #TVRAM,a0
112:       move.w  #SAVLINES-1,d1
113: _L2:
114:          bsr      countdata
115:          move.w  d0,(a5)+ * line length
116:          addq.l  #1,d5      * size of data in buffer (word 単位
)
117:          add.l   d0,d5
118:          movea.l a0,a4
119:          bra      _L3
120: _L4:          move.w  (a4)+,(a5)+ * line data
121: _L3:     dbra     d0,_L4
122:          lea.l    NEXTLINE(a0),a0
123:       dbra     d1,_L2
124:
125:       DOS      _SUPER
126:       addq.l   #4,sp
127:       move.l   d5,datasiz
128:       movem.l (sp)+,reg
129:       rts
130:
131: loadtvram: * (workptr)~ から Text VRAM に復帰する
132: reg = d1/a0-a1
133:       movem.l reg,-(sp)
134:       clr.l    -(sp)
135:       DOS      _SUPER
136:       move.l   d0,(sp)
137:       move.w   CRTC_R21,-(sp)
138:       move.w   #$0130,CRTC_R21 * テキストプレーン 0 & 1 同時アクセス
139:
140:       move.l   #TVRAM,a0
141:       move.w   #SAVLINES-1,d1
142: _L12:
143:          move.w  (a5)+,d0      * データ長の取り出し
144:          movea.l a0,a1
145:          bra      _L13
146: _L14:         move.w  (a5)+,(a1)+ * ビットイメージの転送
147: _L13:    dbra     d0,_L14
148:          lea.l    NEXTLINE(a0),a0
149:       dbra     d1,_L12
150:
151:       move.w   (sp)+,CRTC_R21
152:       DOS      _SUPER
153:       addq.l   #4,sp
154:       movem.l (sp)+,reg
155:       rts
156:
157: _start:
158:       DOS      _VERNUM
159:       cmpi.w   #$0200,d0
160:       bcs      errver
161:       move.w   d0,sysver
162:
163:       CHKARG
164:
165:       suba.l   a0,a1
166:       move.l   a1,-(sp)
167:       pea.l    $10(a0)
168:       DOS      _SETBLOCK
169:       addq.l   #8,sp
170:
171:       move.l   #WORKSIZ,-(sp)
172:       DOS      _MALLOC * 一時バッファの確保
173:       addq.l   #4,sp
174:       tst.l    d0
175:       bmi      errmem
176:       move.l   d0,workptr
177:       movea.l  d0,a5
178:
179: * (save screen)
180:       moveq.l #-1,d1
181:       IOCS     _B_LOCATE
182:       move.l   d0,(a5)+ * カーソル位置書き込み
183:       move.l   #2,datasiz * datasiz = 2 words
184:       IOCS     _B_CUROFF
185:       bsr      savetvram * テキスト画面保存
```

```
186:      IOCS     _B_CURON
187:
188: * (一時バッファから保存用バッファにデータ転送.
189:      move.l  datasiz,d0
190:      add.l   d0,d0
191:      move.l  d0,-(sp)  * datasiz * 2 bytes
192:      move.w  #2,-(sp)  * メモリの上位から
193:      DOS_S_MALLOC      * 保存用バッファを確保
194:      addq.l  #6,sp
195:      tst.l   d0
196:      bmi     errmem2
197:      move.l  d0,d7
198:
199:      movea.l workptr,a3  * source
200:      movea.l d7,a4       * dest
201:      move.l  datasiz,d2  * size (word 単位)
202:      bra     _L6
203: _L5    move.w  (a3)+,(a4)+
204: _L6 dbra d2,_L5
205:
206:      move.l  workptr,-(sp)
207:      DOS     _MFREE       * テンポラリバッファの開放
208:      addq.l  #4,sp
209:      move.l  d7,workptr
210:
211: * (execute command)
212:      clr.l   -(sp)     * 環境(自分と同じ)
213:      pea.l   cmdlin
214:      pea.l   1(a2)     * 起動するファイルネーム
215:      move.w  #2,-(sp) * コマンドラインの展開
216:      DOS     _EXEC
217:      tst.l   d0
218:      bmi     errexec2
219:
220:      clr.w   (sp)      * ファイルの起動
221:      DOS     _EXEC
222:      lea.l   14(sp),sp
223:      tst.l   d0
224:      bmi     errexec0
225:      move.w  d0,exitcode * プロセス終了コードの保存
226:
227: * (load screen)
228:      moveq.l #2,d1
229:      IOCS    _B_CLR_ST   * 画面クリア
230:
231:      movea.l workptr,a5
232:      move.w  (a5)+,d1     * カーソル位置取り出し
233:      move.w  (a5)+,d2
234:      IOCS    _B_LOCATE
235:
236:      IOCS    _B_CUROFF
237:      bsr     loadtvram    * 画面復帰
238:      IOCS    _B_CURON
239:
240:      move.l  workptr,-(sp)
241:      DOS     _MFREE       * 画面の保存用バッファ解放
242:      addq.l  #4,sp
243:      move.w  exitcode,-(sp)
244:      DOS     _EXIT2       * 子プロセスの終了コードを持って終了
245:
246:  .data
247:  .even
248: workptr:    ds.l    1   * バッファのアドレス
249: datasiz:    ds.l    1   * 消費したバッファの量(word 単位)
250: sysver:     ds.w    1   * Human68k のバージョン
251: exitcode:   ds.w    1   * 子プロセスの終了コード
252: cmdlin:     ds.b    256 * コマンドライン展開用バッファ
253:
254:  .end _start
```

今月はショートプロで掲載したMOUSEP.Xの作者の伊藤さんからの質問です。

「MOUSEPはメモリを無駄遣いをしていますが小さいことは気にせずいきましょう。ところでメモリを必要に応じてチビチビ確保する方法と，マウスカーソルをスプライトより優先度を先にする方法を教えてください」だそうです。

小さいことは気にせず，って自分が気にしてるんじゃん(苦笑)。

確かに，Cやアセンブラでプログラムを作っているとメモリを少しずつ確保したいことがありますよね。たとえば一時的にワークエリア用の変数の領域がほしいとかね。

それではスプライトのほうはまた今度として，メモリをチビチビ確保する方法を探ってみましょう。

変数を確保する

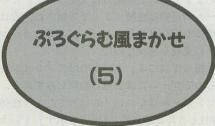
一時的にワークエリア用の変数がほしい。つまり，ある処理を始めるときにメモリを確保して処理が終わったらメモリを開放する。

Cでプログラムを書くときには，いい方法があるんですよね。それがローカル変数。X-BASICにもありますが，これは関数のなかでだけ使える変数で，たとえば関数を書くときに，

```
foo()
{
    int i;
        ⋮
}
```

なんて書いておくと，ほかの関数からはｉは使えないけれど，プログラムがfoo()にいるあいだだけｉを使うことができるんですね。このローカル変数ですが，X680x0の場合，スタックから何バイトか拝借してくることで実現しています。ですから，スタックの範囲内であれば，いくらでも使って捨てることができるから便利ですよん。

スタックを確保する

アセンブラの場合，Cのローカル変数みたいに便利なものはありません。というより，なにをやるにも自分で全部やらなきゃいけないのです。でも逆にいえば，制約もないんですけど。

ですから，自分でスタックをいじってしまえばCでいうところのローカル変数を作ることができます。

たとえば，spがいま，ｘの位置を指しているとします。このときに，

```
subq.l #4,sp
```

とすればspはｘ－４の位置にずれます。これから関数を呼んだりするときにはｘ－４の位置からスタックが使われるわけですから，ｘ番地から４バイト分使われないエリアができるってわけです。

ただし，この使い方には欠点もあります。というのもスタックを変えて変数領域を確保するので，サブルーチン内で使われたスタックを元に戻さないで，元のルーチンに戻ると暴走してしまうのです。

68000ではbsrなどの命令でサブルーチンを呼び出すときには，その呼び出し元のプログラムの番地をスタックに確保してからジャンプし，そこからrtsなどで元のルーチンに戻るときにそのスタックの値を見て戻るのです。そのスタックの位置がずれていると，変な値を見てジャンプするわけですから当然暴走してしまいますよね。

OSの力を借りる

さて，自分だけの力でここまできましたけれど，ちょっと不都合がありますね。それはサブルーチン上からグローバルな値を確保したりすることができませんね。

実はこれをするためのファンクションコールがHuman68kには用意されていて，mallocといいます。これはOSが管理しているメモリの中のいくらかをアプリケーションに分けてくれるDOSコールなのです。

たとえば，先ほどのように４バイトの変数領域を取りたいときには，

```
move.l  #4,-(sp)
dc.w    _MALLOC
addq    #4,sp
```

で確保したメモリのアドレスがd0に入ってきます。あとは，

```
move.l  d0,a1
move.l  d2,a1
```

とでもしてやれば先ほどと同じようにd2の値を確保したメモリに入れることができます。ちなみにメモリのコントロールモードを指定するmalloc2というシステムコールもあります。

Cの場合にもこのmallocは使えます。

```
int * addr;
```

でポインタ変数を確保して，

```
addr = malloc((size_t)4);
```

で４バイトの変数領域を確保できます。

```
* addr = 24;
```

で，この領域に24という数字を入れられます。ほかにもCのメモリ確保の方法としては，配列確保用のcalloc()や使用可能な全メモリ領域を確保するallmem()といった関数が用意されています。詳しくは，マニュアルで確認してみてください。

もちろん，これらのメモリはOSから一時的に借りたものですから，どちらもアプリケーションが終了する前にちゃんと返しておかないといけませんよ。

ちなみにアセンブラの場合，先の例だと，

```
move.l d0,-(sp)
dc.w _MFREE
addq.l #4,sp
```

としておけばいいでしょう。Cの場合には，

```
free(addr);
```

ですね。

……こんなもんでよろしいかな？

では，また来月。

こちらシステムX探偵事務所

FILE-XIV

裏マップエディタを作る

Shibata Atsushi 柴田 淳

実際のピンボールでは自然と決まりますが，コンピュータで再現しようとする場合には手のかかる部分が壁などによる反射です。そんな処理を助けるもののひとつが，裏マップ。今回は裏マップを作るためのツールを作成します。

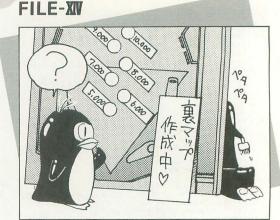
illustration : T.Takahashi

パソコンでピンボールゲームを作るなんて大口を叩いていれば当然だが，ゲームセンターに立ち寄ると必ず本物のピンボールゲームをやることにしている。こうやってゲームの制作記事を書き始める前はピンボールに関してはまったく無知だったのが，何百枚と百円玉を注ぎ込んだかいあって，最近やっとピンボールの面白さの「モト」が見えてきた気がする。

ひと言でいって，ピンボールというのはアメリカ的な球技の代表のひとつである。アメリカというのは面白い国で，外から輸入したスポーツを独自のやり方で別物にしてしまう。イギリス紳士の高貴なお遊びであったクリケットは，ミットやスパイク，あるいはキャッチャーの着けるプロテクターなどのゴツイ器具が加わって野球に進化した。ラグビーはアメフトに，アメリカインディアンが棒の先に網をつけて楽しんでいた素朴な球遊びはラクロスにと，とにかくアメリカナイズされたスポーツというのは道具を使う。アメリカ人というのは道具が大好きな人種なのではないだろうか。

詳しくは知らないのでこれは推測なのだが，ピンボールというのはもともと，ヨーロッパの貴族たちが「オホホ」なんて笑いながら遊んでいた，素朴なスマートボールのようなものに原形があるのだと思う[注)]。で，それがアメリカに取り込まれた途端，いままで重力と摩擦力に支配されていたボールの運動は機械によって増幅され，音は鳴るし電飾は光る，といったメカのカタマリに変貌を遂げたのだ。

「狙って当てる」というのはアクション性のあるゲームには共通した楽しみである。ピンボールはこれに「的に当てたとき機械が(ときには連鎖的に)反応を示す」という要素が加わる。フリッパーをポコポコと叩いているだけでさまざまな反応を示すピンボールの台は，そばで見ているだけでも楽しいものだ。

ただ，台に配置された仕掛けも，あまり簡単に作動しては面白くない。そこで，狭い隙間にボールを通してはじめて作動する仕掛けを設ける。あるいは，点数の高いボーナスを難しい位置に置いたり，連鎖的に仕掛けを作動させなければ発生しないイベントを配するわけである。

編集部注)実際は19世紀アメリカの「バガテル」という現在のビリヤードに似たゲームで，丸い玉を木製のキューで突き，点数を表示した穴に入れるものがもとになっている

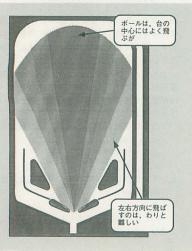
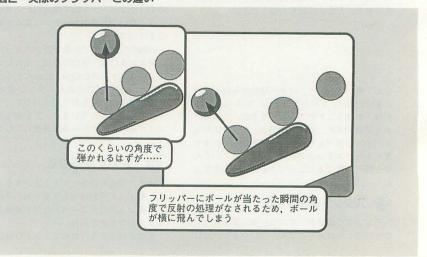
よりリアルなフリッパーを目指す

ピンボールをいろいろ見ていると，台ごとにフリッパーの動く速さが違うことに気づく。またフリッパーの角度も微妙に調整してあり，このあたりから台ごとの個性がにじみ出てくる。

だがたいていの台では，ボールは台の上方向へ鋭角の範囲には弾かれやすいが，台の左右にはなかなか飛んでいかないように統一されている。図1を見てほしい。濃く塗られた扇形の範囲にはボールはよく飛ぶが，扇形の角度が左右に開くにつれて，ボールが弾かれる頻度は少なくなっていく。「難しい位置に面白い仕掛けを置く」というセオリーでいくと，ボールが当たるだけで作動するような仕掛けはピンボール台の左右に配置される，ということになる。

ここで，6月号のフリッパーの処理を思

図1 フリッパーに弾かれるボールの方向

図2 実際のフリッパーとの違い

い出してほしい。フリッパーにボールが当たったことがわかると，反射の処理を行う。つまり，ボールが当たった「瞬間」に，フリッパーがとっている角度の垂直方向にボールは押し出されるわけである。すると，ボールがスリングショット(フリッパーの近くにある三角形のバンパー)の横にあるレーンを通り抜けてきた場合，フリッパーに弾かれたボールは図2の右のように台の横方向に飛んでいく。

ボールがいつも横方向に弾かれるなら，仕掛けを中心に置くように台のレイアウトを変更すればいい。しかし，ボールが直接フリッパーに向かってきて弾かれる場合は，フリッパーの角度は水平に近いことが多く，ボールは台の中心に弾かれてしまうのだ。これではボールを比較的自由な方向に飛ばしやすくなってしまい，都合が悪い。

本物のピンボールを観察した結果，ボールの速度が遅い状態でフリッパーに弾かれる場合は，図2の左のようにボールはフリッパーとの接触面にいったん持ち上げられるようだ。そこでフリッパーの角度が水平になったあたりでやっとフリッパーから離れ，ビヨーンと飛んでいくようである。逆にボールがフリッパーに向かって飛んできたときに弾かれる場合は，ボールはあまり持ち上げられない。要するに，ボールはフリッパーの向いている方向に弾かれるので，このようなときは横方向に弾かれやすい(ただし，フリッパーを押し出すタイミングはかなり厳しい)。

そこで，「フリッパーがボールを持ち上げる」という処理をどうにかしなければならないのだが，けっこうやっかいである。6月号の繰り返しになるが，フリッパーは滑らかに回転するのではなく，飛び飛びに回転角を変えていく。またフリッパーの根元にボールが当たる場合と，先端に当たる場合では，ボールの持ち上げられ方に差が出てくる(先端に当たるほうがボールは長く持ち上げられる)。

フリッパーが動くとき，角度は一定で増えていくと仮定すると，「持ち上げられ具合」を算出することができる。フリッパーの中心軸とボールの中心の距離を，1フレームに動くフリッパーの角度分のsin，cosに掛け合わせれば，ボールがフリッパーに持ち上げられる距離が出てくる。だがこの方法でいくと，距離を算出する時点でどうしても実数を扱わなければならない。これはあまりスマートな方法ではない気がする。それに，個人的には実数を使うような方法はなるべく避けたい。

さて，裏マップのフリッパーの周りには，実際のフリッパーの幅より広い範囲にフリッパーがあることを示す情報が書き込まれている，ということを思い出していただきたい。ボールの中心に当たるマップ情報を拾い出してフリッパーがあることがわかると，反射の処理を行う。これは，ボールがフリッパーをすり抜けてしまう現象を防止するための措置であった。

いろいろ試してみた結果，図3の網のかかった領域にボールがあるうちは，少しずつボールの移動ベクトルに力を加え続けていくような処理をすると，ボールの弾かれ方はかなり自然になることがわかった。

裏マップでは，移動速度の速いフリッパーの外側のマップを，図3のように広く描き出している。つまり，フリッパーの根元にあるときより先端にあるときのほうが，ボールはより長い時間，図3の網のかかった領域に入っているので，それだけ長い間力を受けることになり，ボールの速度は増す。フリッパーの根元に当たったときはボールはそれほど強く弾かれないが，先端に当たったときは大きな力を受けることが直感的にわかるだろう。フリッパーに当たった場所によりボールの弾かれ方が変わるという現象も，この方法で一緒に表現できてしまうわけで，なんともおトクな解決策である。

裏マップエディタの基本設計

さてさて，今月は裏マップエディタを作るのであった。制作中のピンボールでは，台の1ドットごとに1バイトを割り当てた「裏マップ」というのを使っている。このマップには，対応するドットに何があるか，またそこに壁があるのならその壁の角度などの情報を書き出しておき，ボールの移動に合わせて随時参照して，ボールの挙動を決定する。この裏マップを作るツールを作ってみようというのである。

だが考えてみると，たかだかゲームを制作するのにいちいちエディタから作るというのは，いってみれば大工が家を建てるのに，ノコギリやカンナから作り始めるのに似て馬鹿げているように思える。しかし実際は，512×512の画面全体で256Kバイトにも及ぶ裏マップを手作業で編集することに比べたら，エディタを作るほうがずっと効率的だ。

と，ひとしきり講釈の終わったところで，今回作る裏マップエディタの基本設計を決めよう。まず，画面は裏マップと台のラフデザインとを重ね合わせて拡大表示する。したがって当然画面に表示されるのは全体でなく一部分だけとなるので，表示されていない部分を編集するときには，画面をスクロールさせることになる。

編集画面上では，壁の角度がどのくらいか，視覚的にわかったほうが都合がよい。いままでのサンプルでは，裏マップを自動生成する関係上，壁の角度は0度から180度まで60分割されていたが，これからは36分割とする。この36種類の壁の傾き具合を，8×8のパターンで表現しよう（このパターンは自前で用意しなければならない）。

ピンボール台の上には，ボールを跳ね返す壁だけでなくいろいろな仕掛けも配置することになる。8ビットの裏マップでは0から255までの数字が表現可能だから，まず0をなにもない場所，1から36までを壁に割り当てる。で，37からはこれらの仕掛けの存在を示すコードに割り当てる。壁の角度は表示されるパターンで表したが，この仕掛けに関しては単純に16進数を表示するだけにする。

エディタの仕様が決まると，さしあたって必要なのは傾いた壁と0からFまでの文字のパターンだ。傾いた壁のパターンは，X-BASICで10度刻みに回転していくラインのビットマップを出力する簡単なプログラムを組んでファイルに出力し，エディタで体裁を整えてリストに取り込んだ。リストの22行目からがそのデータである。ただし実際にリストに載せてあるのは36種類のパターンのうち18種類だけで，残りはすで

図3　ボールが力を受ける範囲

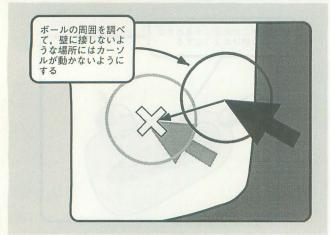
にあるものを垂直反転して流用している。

壁のパターンの下に並ぶのは8×8の文字のパターン。これはずいぶん昔からあるクレヨンというフリーウエアを使用した。クレヨンは白黒のお絵描きソフトで，タイトル形式でファイルを出力すればビットマップのデータが得られる。これをダンプして壁のパターンと同様に体裁を整え，リストに取り込んだ。

なお，パターン関係のデータはすべて構造体として定義されている。IOCSコールのTEXTPUTを使い，パターンをX680x0のテキスト画面に表示するためにこうしたものだ。TEXTPUTの要求するデータ構造はワード長のX，Yのパターンサイズと，それに続く任意長のバイトデータである。11，12行にある構造体の定義を見れば納得していただけるだろう。

表示に必要なパターンが揃ったら，次は拡大描画のルーチンを作る。ただし画面をスクロールさせる関係上，表示はできるだけ速くしたい。グラフィック画面でなくテキスト画面を使ったのはそのためもあるのだが，それでも768×512の画面全体を描き換えるのは非常に骨が折れる。

描き換え時になるべく無駄をはぶく，という意味合いから，次のような工夫をしよう。画面に表示されるパターンは8×8のサイズだから，画面全体には96×64のマス目が張り巡らされていることになる。そこで，このサイズ分の配列をひとつ用意する。最初に画面を描くときは，この配列に描かれるパターンの番号を保存しておく。2回目以降は，描くべきパターンと直前に描かれているパターン番号(つまり配列の内容)を比較して，同じなら描き換えない。

こうすることで必要最小限の部分だけを描き換えることができるようになるわけだ。知っていると知らないとでは天と地ほどの差がある技法だが，非常に古典的な方法論でもある。あんまり古くさい技法ばっかり解説しているのは気が退けるので，こんなものはすっ飛ばして先を急ごう。

編集効率を上げる

ゲーム作成の効率を上げることを目的として作るこの裏マップエディタも，画面表示周りの関数を作ってしまえば，あとはマウスカーソルのある場所のマップ情報を描き換える機能をつけるだけでエディタとしての体裁は一応整う。ただ，これだけではいかにも低機能である。そこで，編集作業の効率を上げる機能について少々検討してみたい。

エディタでは壁の角度を編集するわけだが，この壁の情報というのはどのような位置に描かれていただろうか。5月号を思い出すと，壁からちょうどボールの半径分だけ離れた位置に描き出されたのであった。こうすることによって，ボールと壁との接触を判定するとき，ボールの周囲をすべて調べるのではなく中心だけを調べればよくなるのだ。

で，この「壁からボールの半径分だけ離れた位置」というのが厄介である。ボールの半径を4ドットとすると，壁が垂直か水平なら単純に半径のドット数だけ離れた場所に壁の角度情報を描けばいい。しかし，壁が斜めになっているときなどは，一見しただけではどこに壁の角度情報を描けばいいか判断がつかない。

こういう単純作業を人間側に強いるのはあまり美しくない。そこで，マップを編集中に，壁の情報を描き出すべきかどうか判断する機能を設けよう。

具体的な方法としては，まず，マウスカーソルをボールの中心に見立てる。またボールの周囲にあたる位置の座標情報を，あらかじめ配列に入れておく。マウスカーソルが移動するたびに周囲にあたるドットをすべて調べ，もし壁が見つからなかったらマウスカーソルの位置を元の場所に押し戻す。

こうすれば壁の情報が描き込まれないはずの位置にはマウスカーソルがいかなくなるので，編集効率は増すはずだ。ただし，実際には壁のない位置にもバランスの微調整などの理由から壁の角度情報を描きたいときもあるだろうから，壁のない場所にはマウスカーソルが動かない「ロック」モードと，カーソルをどこにでも自由に移動できる「ロック解除」の2つのモードを設けるようにする。

図4　ボールの周囲に壁が見つからない場合

マップの自動生成

このようにして，壁の情報を描き出す位置がわかるなら，あとは壁の角度を自動的に計算するような方法が見つかれば，マップを自動的に作れるようになる。そうすれば，あとはペイントツールなどで適当に台のデザインをしてこのマップエディタに放り込むだけで，実際にボールを弾いて台のバランステストができてしまう(仕掛けなどは別に作らなければならないが)。

台の角度を求める方法としてひとつ考えられるのが，壁の情報をベクトルとして別に編集する，という方法だ。ただし，これでは余分な情報を編集する手間が増えてしまうので「マップの自動生成」とはいい難い。

余談だが，パソコンなどのピンボールゲームには，このように壁の情報をベクトルとして書き出してボールとの接触判定をするものがあるようである。この場合は，この記事で作っているようにボールを1ドットずつ動かさなくても済む，という利点がある。そのかわりに，ボールが動くたびに登録されたベクトルとボールの軌跡の，交差判定をしなければならない。

交差判定自体は，それほど重たい処理ではない。ただし，いちいちたくさんのベクトルとの交差判定をするのは無駄だろうから，あらかじめリストなどを作っておいて，最低限の交差判定をすれば済むような工夫は必要だろう。結局は僕の採用した裏マップを用意する手法と比べて格段に簡単な(あるいは軽い)処理とはいい難い。またその逆もいえる。

余談ついでに，Macintoshのとあるピンボールゲームは，なんとコプロセッサがないと動かない。アクション性のあるゲームに浮動小数点演算が必要なんて，3Dモノならいざ知らず，ちょっとびっくりする。

話を元に戻そう。ベクトルを別に書き出す方法はとりあえずパス。次の候補としては，問題となる裏マップ上の1点の周りの広い範囲を調べ，壁の「縁」を抽出し，縁の点の並びから壁の角度を類推する，という方法がある。ただし，これも512×512の

領域すべてを調べなければならないとなると，けっこう時間がかかりそうだ。

ここで，前の節で説明した「ボールの周囲にあたる位置の座標」のことを思い出していただきたい。ある点が壁に触れているとすると，その壁のボールの中心から見た角度もわかるはずだ。たとえば，ボールの左の点に壁が接していれば，壁の角度は180度となる(裏マップ内の表現では0度と同じ)。

ボールの周りのひとつの点だけが壁に接しているのであれば，これで壁の角度を特定できる。問題は，ちょうど図4のようにたくさんの点が壁に接している場合である。では，いっそのことボールの周りのたくさんの点が壁に接しているときは，接している部分の角度を平均化してしまったらどうだろうか。

なんともあっけないが，これで裏マップの自動生成ができてしまう。サンプルプログラムでは，画面のすべての点について壁があるかどうか調べ，見つかった場合は角度を平均し，壁の角度の情報をマップに描き出している。また，調べている点上に壁があるなら，そこには壁の角度の情報は描かれないはずだ。このような場合は周囲の点を調べる処理をしないようにし，処理の高速化を図っているが，それでもひとつのマップを自動生成するのに数分かかる。台のデザインというのはたいてい真ん中がぽっかり開いているから，範囲を指定して必要なところだけの裏マップを自動生成するようにしたほうがよりスマートになるだろう。

裏マップエディタについて

実際に僕が裏マップの編集に使っているものは，サンプルよりも少々高機能である。リスト掲載の関係から，冗長な（とはいえ僕自身にとっては欠くべからざる）機能は削ってしまった。が，サンプルの裏マップエディタは，一応必要最低限の機能を備えている。

まず，マウスを使ってカーソルを動かす。右クリックをすることによりカーソルに描かれているパターンがマップに置かれる。また，ROLL UP，DOWNのキーを押すと，カーソルのパターンを変えることができ，スペースキーを押すとカーソルの位置にあるパターンを取り込む。

ボールが壁に接しない位置にカーソルを動かさないようにするロック機能は，TABキーで働く。もう一度TABキーを押すと，ロック機能が解除される。また，Wキーを押すとマップの自動生成が始まる。始まると数分は止まらないので，注意していただきたい。

せっかくマップエディタを使って作った裏マップも，実際に作った台の上でボールを動かせなければちっとも面白くない。反射の処理などは先月号の関数reflectをそのまま流用できるし，先月号のサンプルで壁の角度が60分割になっていたのを36分割にするだけで，今回のマップエディタの吐き出した裏マップを使うことができる。

来月号では，本題のリストの分量いかんによっては，サンプルのエディタで編集した裏マップ上でボールを動かすことのできる簡単なサンプルを掲載させていただくかもしれない。そして本題は，ピンボール台を2階建て構造にすることを検討するつもりだ。具体的には，ボールのパターンのクリッピングとか，坂道の表現などの処理を考えたいと思っている。　(つづく)

リスト

```
  1: /* 裏マップエディタ */
  2: #include <iocslib.h>
  3: #include <doslib.h>
  4: #include <stdlib.h>
  5: #include <stdio.h>
  6: #include <basic.h>
  7: #include <basic0.h>
  8: #include <graph.h>
  9: char scr[6144];
 10: unsigned char tsc[6144];
 11: char bmp[512][512];
 12: struct { UWORD x,y;
 13:       char f[8];
 14:     } edpat = {
 15:     8,8,
 16:     0,0,0,0,0,0,0,0 },
 17:     edfil = {
 18:     8,8,
 19:     0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff },
 20:     edclr = {
 21:     8,8,
 22:     0,0,0,0,0,0,0,0 },
 23:     slp[37] = {
 24:     8,8,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
 25:     8,8,0x00,0x00,0x00,0xff,0x00,0x00,0x00,0x00,
 26:     8,8,0x00,0x00,0xc0,0x3f,0x00,0x00,0x00,0x00,
 27:     8,8,0x00,0x00,0xe0,0x1f,0x00,0x00,0x00,0x00,
 28:     8,8,0x00,0x00,0xe0,0x1e,0x01,0x00,0x00,0x00,
 29:     8,8,0x00,0x00,0xe0,0x1c,0x03,0x00,0x00,0x00,
 30:     8,8,0x00,0x80,0x60,0x18,0x06,0x01,0x00,0x00,
 31:     8,8,0x00,0x00,0xc0,0x30,0x0c,0x02,0x01,0x00,
 32:     8,8,0x00,0xc0,0x20,0x10,0x0c,0x02,0x01,0x00,
 33:     8,8,0x80,0x40,0x20,0x10,0x08,0x06,0x01,0x00,
 34:     8,8,0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01,
 35:     8,8,0x80,0x40,0x20,0x10,0x08,0x04,0x04,0x02,
 36:     8,8,0x40,0x40,0x20,0x10,0x08,0x08,0x04,0x02,
 37:     8,8,0x40,0x20,0x20,0x10,0x10,0x08,0x04,0x04,
 38:     8,8,0x40,0x20,0x20,0x10,0x10,0x08,0x08,0x04,
 39:     8,8,0x20,0x20,0x20,0x10,0x10,0x10,0x08,0x08,
 40:     8,8,0x20,0x20,0x20,0x10,0x10,0x10,0x10,0x08,
 41:     8,8,0x20,0x20,0x20,0x10,0x10,0x10,0x10,0x10,
 42:     8,8,0x20,0x20,0x20,0x10,0x10,0x10,0x10,0x10,
 43:     8,8,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,
 44:     },
 45:     fgp[37] = {
 46:     4,8,0x00,0x70,0x50,0x50,0x50,0x70,0x00,0x00,
 47:     4,8,0x00,0x20,0x60,0x20,0x20,0x70,0x00,0x00,
 48:     4,8,0x00,0x70,0x10,0x70,0x40,0x70,0x00,0x00,
 49:     4,8,0x00,0x70,0x10,0x70,0x10,0x70,0x00,0x00,
 50:     4,8,0x00,0x50,0x50,0x70,0x10,0x10,0x00,0x00,
 51:     4,8,0x00,0x70,0x40,0x70,0x10,0x70,0x00,0x00,
 52:     4,8,0x00,0x40,0x40,0x70,0x50,0x70,0x00,0x00,
 53:     4,8,0x00,0x70,0x10,0x20,0x20,0x20,0x00,0x00,
 54:     4,8,0x00,0x70,0x50,0x70,0x50,0x70,0x00,0x00,
 55:     4,8,0x00,0x70,0x50,0x70,0x10,0x10,0x00,0x00,
 56:     4,8,0x00,0x20,0x50,0x70,0x50,0x50,0x00,0x00,
 57:     4,8,0x00,0x60,0x50,0x60,0x50,0x70,0x00,0x00,
 58:     4,8,0x00,0x70,0x50,0x40,0x50,0x70,0x00,0x00,
 59:     4,8,0x00,0x60,0x50,0x50,0x50,0x60,0x00,0x00,
 60:     4,8,0x00,0x70,0x40,0x70,0x40,0x70,0x00,0x00,
 61:     4,8,0x00,0x70,0x40,0x70,0x40,0x40,0x00,0x00
 62:     },
 63:     ppat[16] = {
 64:     8,8,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
 65:     8,8,0x01,0x44,0x01,0x44,0x01,0x44,0x01,0x44,
 66:     8,8,0x25,0x52,0x25,0x52,0x25,0x52,0x25,0x52,
 67:     8,8,0xaa,0x55,0xaa,0x55,0xaa,0x55,0xaa,0x55,
 68:     8,8,0xad,0x55,0xad,0x55,0xad,0x55,0xad,0x55,
 69:     8,8,0x57,0x75,0x57,0x75,0x57,0x75,0x57,0x75,
 70:     8,8,0xee,0xbb,0xee,0xbb,0xee,0xbb,0xee,0xbb,
 71:     8,8,0xdf,0xfd,0xdf,0xfd,0xdf,0xfd,0xdf,0xfd,
 72:     };
 73: struct { UWORD x,y;
 74:     } bedge[45] = {
 75:     14,7, 14,6, 13,5, 13,4, 12,3, 12,2, 11,2, 10,1,
 76:     9,1, 8,0, 8,0, 7,0, 6,0, 5,1, 4,1, 3,2, 2,2, 2,3,
 77:     1,4, 1,5, 0,6, 0,7, 0,8, 0,8, 1,9, 1,10, 2,11,
 78:     2,12, 3,12, 4,13, 5,13, 6,14, 7,14, 8,14, 8,14,
 79:     9,13, 10,13, 11,12, 12,12, 12,11, 13,10, 13,9,
 80:     14,8, 14,8 };
 81:  int br[45] = { 16,14,12,11,10,8,6,4,2,0, 72,71,69,68,67,66
,
 82:     64,63,62,60,59,57,55, 54,52,50,48,46,44,42,43,41,39,37,
 83:     36,34,32,29,27,25,22,20,18,18 };
 84:  int mx,my=0,px=1,py=0,p,b,c = 0,i,j,l,pud = 0,puc = 0,
 85:     rf = 0;
 86:  unsigned char k[16],tmp[32],pat = 0,pp = 1;
 87:  int x=0,y=0,ox=0,oy=0,cmod = 1,dmod = 0,emod = 0,
 88:     quit = 0,ppx,ppy;
 89:  int x1=0,y1=0,x2=0,y2=0;
 90:  int pe,p1,p2,p3,p4;
 91:  int rx1,rx2,ry1,ry2;
 92:  double r = 64;
 93:  struct FILE *fp;
 94: void main()
 95: {
 96:     init();
 97:     txfill( 0,0,767,511 );
 98:     while( main_loop() == 0 );
 99:     mouse( 2 );
100:     txfill( 0,0,767,511 );
101:     fp = fopen( "bmap","wb" );
102:     if( fp != NULL )
103:     {
104:         i = fwrite( &bmp,4,65536,fp );
105:         fclose( fp );
106:     }
107:     OS_CURON();
```

```
108:    while( kbhit() )
109:        getch();
110: }
111:
112: int main_loop()
113: {
114:  int ppp;
115:    rf = 0;
116:    mspos( &mx,&my );
117:    msstat( &i,&i,&b,&c );
118:    ppp = pat;
119:    key_opr();
120:    if( pat != pp )
121:    {
122:        if( pat < 37 )
123:        {
124:            for( j = 0; j < 8; j++ )
125:                edpat.f[j] = slp[pat].f[j];
126:        }
127:        else
128:        {
129:            for( j = 0; j < 8; j++ )
130:                edpat.f[j] = fgp[pat/16].f[j] |
131:                    ( fgp[pat & 15].f[j] >> 4 );
132:        }
133:    }
134:    if( ox != (mx & 0xff8) || oy != (my & 0xff8) || pat != p
p )
135:    {
136:        l = 0;
137:        if( emod == 1 )
138:        {
139:            if( pat < 37 )
140:            {
141:                for( j = 0; j != 44; j++ )
142:                {
143:                    i = point( mx/8+x+bedge[j].x-7,
144:                        my/8+y+bedge[j].y-7 );
145:                    if( i != 2 && i != 0 )
146:                    {
147:                        l = 1;
148:                        break;
149:                    }
150:                }
151:            }
152:            if( pat >= 37 )
153:            {
154:                for( j = 0; j != 44; j++ )
155:                {
156:                    i = point( mx/8+x+bedge[j].x-7,
157:                        my/8+y+bedge[j].y-7 );
158:                    if( i != 2 && i != 0 && i != 1 )
159:                    {
160:                        l = 1;
161:                        break;
162:                    }
163:                }
164:            }
165:        }
166:        else
167:            l = 1;
168:        if( l == 1 )
169:        {
170:            window( 0,511,1023,1023 );
171:            if( pat < 37 && cmod != 0 )
172:            {
173:                fill( ox-63,oy+446,ox+68,oy+580,0 );
174:                draw_guide( 0 );
175:            }
176:            mouse( 2 );
177:            put_edp( &edclr,&edclr );
178:            ox = mx & 0xff8;
179:            oy = my & 0xff8;
180:            put_edp( &edpat,&edfil );
181:            if( pat < 37 && cmod != 0 )
182:            {
183:                circle( ox+3,oy+515,64,5,0,360,256 );
184:                if( pat != 0 )
185:                {
186:                    j = (pat+17)%36;
187:                    x2 = cos((pat-1)/36.0*pi())*r;
188:                    y2 = sin((pat-1)/36.0*pi())*r;
189:                    x1 = cos(j/36.0*pi())*r;
190:                    y1 = sin(j/36.0*pi())*r;
191:                    draw_guide( 5 );
192:                }
193:            }
194:            mouse( 1 );
195:            window( 0,0,1023,1023 );
196:        }
197:        else
198:        {
199:            setmspos( ox+4,oy+4 );
200:        }
201:    }
202:    pe = pat;
203:    if( b != 0 )
204:    {
205:        bmp[oy/8+y][ox/8+x] = pat;
206:        t_draw( x,y );
207:    }
208:    if( px != x || py != y )
209:    {
210:        rf = 1;
211:        if( dmod == 0 )
212:        {
213:            g_lupe( x,y );
214:            t_draw( x,y );
215:        }
216:    }
217:    if( ppp < 37 && pat > 36 )
218:    {
219:        fill( ox-63,oy+446,ox+68,oy+580,0 );
220:        draw_guide( 0 );
221:    }
222:    px = x;
223:    py = y;
224:    pp = pat;
225:    indcate();
226:    return( quit );
227: }
228:
229: void key_opr()
230: {
231:    int i,j;
232:    for( i = 0; i < 16; i++ )
233:        k[i] = BITSNS( i );
234:    p = 2;
235:    if( k[0] & 2 )
236:        quit = 1;
237:    if( k[14] & 4 )
238:        p = 128;
239:    if( k[14] & 1 )
240:        p = 32;
241:    if( k[7] & 8 )
242:        x -= p;
243:    if( k[7] & 32 )
244:        x += p;
245:    if( k[7] & 16 )
246:        y -= p;
247:    if( k[7] & 64 )
248:        y += p;
249:    if( x > 448 )
250:        x = 448;
251:    if( y > 448 )
252:        y = 448;
253:    if( x < 0 )
254:        x = 0;
255:    if( y < 0 )
256:        y = 0;
257:    if( k[7] & 1 && pat < 37 && pud != 1 )
258:    {
259:        pat++;
260:        pat %= 37;
261:        if( pat == 0 )
262:            pat = 1;
263:        pud = 1;
264:        puc = 0;
265:    }
266:    if( k[7] & 2 && pat < 37 && pud != -1 )
267:    {
268:        pud = -1;
269:        puc = 0;
270:        if( pat != 1 )
271:        {
272:            pat--;
273:            pat %= 38;
274:        }
275:        else
276:            pat = 36;
277:    }
278:    if( k[7] & 1 && pat > 36 && pud != 1 )
279:    {
280:        pat++;
281:        pud = 1;
282:        puc = 0;
283:    }
284:    if( k[7] & 2 && pat > 36 && pud != -1 )
285:    {
286:        pud = -1;
287:        puc = 0;
288:        if( pat != 37 )
289:        {
290:            pat--;
291:        }
292:    }
293:    puc ++;
294:    puc %= 200;
295:        if( (int)(k[7] & 3) == 0 )
296:            pud = 0;
297:    else
298:    {
299:        if( puc == 0 )
300:            pud = 0;
301:    }
302:    if( k[6] & 32 )
303:        pat = bmp[oy/8+y][ox/8+x];
304:    if( k[2] & 4 )
305:        amap();
306:    if( k[2] & 2 )
307:    {
308:        rf = 1;
309:        if( cmod == 1 )
310:        {
311:            cmod = 0;
312:            window( 0,511,1023,1023
313:            wipe();
314:            window( 0,0,1023,1023 );
315:        }
316:        else
317:            cmod = 1;
318:        kw();
319:    }
320:    if( k[2] & 1 )
321:    {
322:        if( emod == 0 )
323:            emod = 1;
324:        else
325:            emod = 0;
326:        rf = 1;
```

▶就職活動で死にかけております。う～む，とんでもない年に当たったものだ。
鈴木　真一(24)埼玉県

```
327:        kw();
328:     }
329:  }
330:
331: void draw_guide( c )
332: int c;
333:  {
334:     line(ox+3+x1,oy+515+y1,
335:     ox+3-x1,oy+515-y1,c,'NASI');
336:     line(ox+3+x1+x2*2,oy+515+y1+y2*2,
337:     ox+3+x1-x2*2,oy+515+y1-y2*2,c,'NASI');
338:     line(ox+3-x1+x2*2,oy+515-y1+y2*2,
339:     ox+3-x1-x2*2,oy+515-y1-y2*2,c,'NASI');
340:  }
341:
342: void put_edp( p1,p2 )
343: char *p1,*p2;
344:  {
345:     TCOLOR( 8 );
346:     TEXTPUT( ox,oy,p1 );
347:     TCOLOR( 4 );
348:     TEXTPUT( ox,oy,p2 );
349:     TCOLOR( 1 );
350:  }
351:
352: void indcate()
353:  {
354:     if( rf != 0 )
355:     {
356:        locate( 0,0 );
357:        if( emod == 1 )
358:            printf( "Lock on  (TAB) :" );
359:        else
360:            printf( "Lock off (TAB) :" );
361:        if( cmod == 1 )
362:            printf( "Guide on  (Q) :" );
363:        else
364:            printf( "Guide off (Q) :" );
365:     }
366:     if( oy/8+y != ppy || ox/8+x != ppx )
367:     {
368:        locate( 0,1 );
369:        printf( "Position ( %3d : %3d )",
370:            ox/8+x,oy/8+y );
371:     }
372:     ppy = oy/8+y;
373:     ppx = ox/8+x;
374:  }
375:
376: void g_lupe( x,y )
377: register int x,y;
378:  {
379:   register int i,j,xx,px,yy,k = 0;
380:   char pl = 0,n,pn = 0;
381:   char ll;
382:     TCOLOR(2);
383:     for( i = 0; i < 64; i++ )
384:     {
385:        xx = x;
386:        yy = i*8;
387:        for( j = 0; j < 96; j++ )
388:        {
389:            ll = (char)point( xx,y );
390:            if( ll != scr[k] )
391:            {
392:                px = j*8;
393:                TEXTPUT( px,yy,&ppat[ll] );
394:                scr[k] = ll;
395:            }
396:            k++;
397:            xx++;
398:        }
399:        y++;
400:     }
401:     TCOLOR(1)
402:  }
403:
404: void t_draw( x,y )
405: register int x,y;
406:  {
407:   register int i,j,k;
408:   unsigned char tmp[32],l;
409:     for( i = 0; i < 64; i++ )
410:     {
411:        k = i * 96;
412:        for( j = 0; j < 96; j++ )
413:        {
414:            l = bmp[y+i][x+j];
415:            if( tsc[k+j] != l )
416:            {
417:                if( l < 37 )
418:                {
419:                    TEXTPUT( j*8,i*8,&slp[l] );
420:                }
421:                else
422:                {
423:                    TEXTPUT( j*8,i*8,&fgp[l/16] );
424:                    TEXTPUT( j*8+4,i*8,&fgp[l & 15] );
425:                }
426:                tsc[k+j] = l;
427:            }
428:        }
429:     }
430:  }
431:
432: kw()
433:  {
434:   unsigned char k[16];
435:   int i,j = 0;
436:     for( i = 0; i < 16; i++ )
437:            k[i] = BITSNS( i );
438:     while( j == 0 )
439:     {
440:        for( i = 0; i < 16; i++ )
441:            if( k[i] != BITSNS( i ) )
442:                j = 1;
443:     }
444:  }
445:
446: void amap()
447:  {
448:   int i,j,k,l = 0,m,r;
449:     for( i = 0; i != 512; i++ )
450:     {
451:        for( j = 0; j != 512; j++ )
452:        {
453:            r = point( j,i );
454:            if( r != 1 )
455:            {
456:                l = 0;
457:                m = 0;
458:                for( k = 0; k != 44; k++ )
459:                {
460:                    r = point( j+bedge[k].x-7,
461:                        i+bedge[k].y-7 );
462:                    if( r == 1 )
463:                    {
464:                        m += br[k];
465:                        l++;
466:                        if( l > 8 )
467:                        {
468:                            l = 0;
469:                            break;
470:                        }
471:                    }
472:                }
473:                if( l != 0 )
474:                {
475:                    m = (double)(m+0.5)/(double)l;
476:                    bmp[i][j] = (m%36)+1;
477:                }
478:            }
479:        }
480:        locate( 0,1 );
481:        printf( "( %3d )",i );
482:     }
483:  }
484:
485: void init()
486:  {
487:     screen( 2,0,1,1 );
488:     execl( "apic.r","/l","bord.pic",NULL );
489:     cls();
490:     OS_CUROF();
491:     palet( 0,0 );
492:     palet( 5,rgb( 5,31,10 ) );
493:     window( 0,0,1023,1023 );
494:     home( 0,0,512 );
495:     fp = fopen( "bmap","rb" );
496:     if( fp != NULL )
497:     {
498:        fread( &bmp,4,65536,fp );
499:        fclose( fp );
500:     }
501:     for( i = 0; i != 18; i++ )
502:     {
503:        for( j = 0; j != 8; j++ )
504:        {
505:            slp[i+20].f[j] =
506:              slp[18-i].f[7-j];
507:            slp[i+20].x = 8;
508:            slp[i+20].y = 8;
509:        }
510:     }
511:     mouse( 4 );
512:     mouse( 1 );
513:  }
514:
515: void txfill( x1,y1,x2,y2 )
516: int x1,y1,x2,y2;
517:  {
518:   struct TXFILLPTR tf;
519:   int xx,yy;
520:     if( y1 > y2 )
521:     {
522:        yy = y2;
523:        y2 = y1;
524:        y1 = yy;
525:     }
526:     if( x1 > x2 )
527:     {
528:        xx = x2;
529:        x2 = x1;
530:        x1 = xx;
531:     }
532:     tf.vram_page=0;
533:     tf.x=x1;
534:     tf.y=y1;
535:     tf.x1=x2-x1+1;
536:     tf.y1=y2-y1+1;
537:     tf.fill_patn= 0;
538:     TXFILL(&tf);
539:     tf.vram_page=1;
540:     TXFILL(&tf);
541:     tf.vram_page=2;
542:     TXFILL(&tf);
543:     tf.vram_page=3;
544:     TXFILL(&tf);
545:  }
```

▶「セーラームーン」も「ドラゴンボール」なみに続くのでしょうか? 娘は喜んでおりますが……。
松本 弘道(41)佐賀県

SX-BASIC

SX-BASIC公開デバッグ 第5回

エラー処理と百人一首

Ishigami Tatsuya 石上 達也

今回はエラー処理そのほかについて解説し，おまけに禁断のAラインエミュレータ直接呼び出しにも手を出します。サンプルはSX-BASICによるゲーム「タイムトライアル百人一首」です

エラーメッセージ

SX-BASICは，なるべくX-BASIC互換となるように開発してきましたが，大きくX-BASICと食い違っている箇所があります。エラー表示の部分です。

たとえば，

print "Hello World "

とするところを，

prnt "Hello World "

のように間違えてしまった場合，画面には，

「未宣言の変数です」

と表示されてエラーになってしまいます。

理由は簡単で，私がさぼっているからにほかなりません。SX-BASICは後述のように中間コード方式で動作しています。与えられたプログラムの先頭から命令を見ていき中間言語へと変換を行います。変換中に，ん？　と思うとエラー表示を行います。エラーには2種類あって，式の数値がおかしいといった動作時にわかるものと，プログラムの構造，文法がおかしいという翻訳時にわかるものがあります。翻訳時に表示されるということは，つまり文法的に問題のあるプログラムを与えたということです。

では，翻訳時に引っかかるようなものはすべて「文法違反」にしてよいかというと，問題はもう少し複雑で，

int a

a=1+b

のようなプログラムも考えに入れなければならないわけです。

この場合，明らかに「b」は宣言を忘れてしまった変数です。SX-BASICにとっては，宣言されていない変数は意味不明の文字列ということにほかなりませんから，文法エラーになってしまいます。

理由は，

・SX-BASICがプログラムを一方向（左から右，行が変われば上から下）にしか見ない。

・宣言されていない変数は認めない。

といった，頑固な仕様になっているからです。本当は，エラー箇所の前後も総合的に調べ，もっとも適切なエラーメッセージを選択すべきなのですが，SX-BASICはほかにも改良すべき点が山ほどあり，そこまで手が回りません。

文法解析中に発生したエラーはすべて文法違反にしてしまうと，若干問題があり，

Macintoshのこと

Macintoshといえばあらゆるウィンドウシステムのお手本のように思われていますが，完全なマルチタスクシステムになったのは最近のことです。世に出た当時は，ウィンドウは複数開けましたが，一度に実行できるタスクはひとつでした。過去のアプリケーションとの整合性をとるために，いまでも操作体系にその影響を引きずっています。

これは，Macintoshを使ったことのない人にはわかりづらいのですが，

1タスク＝1ウィンドウとは限らない

という状況です。たとえば，テキストファイルのアイコンをダブルクリックします。すると，テキストファイルがダブルクリックされた→テキストファイルを見るにはテキストエディタ，というような関連づけが内部で行われ，テキストエディタが起動されます。起動されたテキストエディタは，先ほどのファイルを「オープン」し，ウィンドウをひとつ開きます（図1.1）。ここまでは，SX-WINDOWとなんら変わりありません。次に，別のテキストファイルをダブルクリックします。すると，この情報は先ほど起動されたエディタに伝えられ，ウィンドウをひとつ開きます（図1.2）。新たにエディタを起動するのではなく，すでに起動されているエディタが，新たにもうひとつウィンドウを開くのです。この状態で，さらに別のテキストファイルをダブルクリックすると，図1.3のような状態になります。

たとえば，複数のテキストウィンドウを開き，3本の文章をいっぺんに書く場合があるかもしれません。このようなときには，ひとつのテキストエディタで，3つのウィンドウを開きます。SX-WINDOWの場合でしたら，テキストエディタのアイコンを3回ダブルクリックすれば準備は完了します。しかし，Macintoshでは，2回目以降のダブルクリックは「タスクの切り換え」という意味（表計算ソフトからドローツールへ，など）ですから，この方法は使えません。

1）　最初のダブルクリックでプログラムを起動。多くのプログラムの場合，ここで「Untitled」とタイトルのついたウィンドウが開きますが，なかにはウィンドウを開かないプログラムも考えられます。とりあえず，ウィンドウが開いても開かなくても，メニューだけはそのプログラムのものになります（SX-WINDOWからはちょっと想像しがたいですが，Macintoshでは基本的にメニューはウィンドウ内に現れるものではなく，デスクトップ上段のメニューバーと呼ばれる専用の領域に現れます）。

2）　メニューから「新規」という項目を選択します。1回選択すると，ひとつウィンドウが開くので，必要な回数だけ選択します。たとえば，図1.1にある状態のテキストエディタから図1.3の状態にするには「新規」を2回選択します。

逆にウィンドウを「閉じる」場合，テキストエディタはクローズボタンが押されたウィンドウ，あるいはメニューから「閉じる」が選択されたウィンドウを閉じます。図1.3のような状態で最後に開いたウィンドウを「閉じる」と図1.2の状態に戻ります。さらにウィンドウを「閉じる」と図1.1の状態に戻ります。さて，最後にウィンドウを閉じるといったいどうなるのでしょう。ウィンドウは閉じても，図2のようにタスク（のプログラム）はメモリ上に残ってしまうのです。次回，テキストファイルを眺めるときには，テキストエディタ本体のロード時間が省略できて便利なのですが，二度と使用しない場合は残ってしまったテキストエディタはただの邪魔ものです。なにも仕事がないくせ，メモリだけは一人前に占拠するのです。

本体をメモリ上から取り除くためにはタスクスイッチ（？）を使って，エディタをアクティブにします。アクティブにするといっても，ウィンドウを持っていないのですから一見なりっこないように思われます。しかし，Macintoshにはメニューバーがウィンドウではなく，デスクトップ上についているので，ウィンドウを持っていないタスクがアクティブになった場合，メニューバーだけ切り換わります（マウスでアク

この箇所で引っかかったらまず未宣言の変数だろうな，と私が思った箇所については，表示するエラーメッセージを変えてあります。この区分はあくまでも，確率的な区分けでしかありません。

そんなわけで，「文法違反」「未宣言の変数です」と表示された場合には，お互い反対側のエラーが発生した可能性もあるわけです。エラーの理由がわからないときにはこのことを思い出してください。

エラー表示が次の行にいってしまう

エラー処理の手抜きによる不具合がもうひとつあります。

以下のプログラムを見てください。

```
10 print "Hello World !!";a
20 print "Hello World !!"
```

SX-BASICでは，使用する変数はすべて宣言されていなければならない，という規則がありましたから，行番号10で使用されている「a」は，未宣言の変数です。したがって行番号10のところで,「未宣言の変数です」エラーが発生するはずです。

しかし，現実には，なぜか行番号20で，エラーが発生します。

理由はいろいろあるのですが，現バージョンのSX-BASICでは，本来表示されるはずの場所より若干後ろにずれた箇所がエラー表示される場合があります。前章での注意とあわせて，原因不明のエラーが発生した場合にはエラー発生箇所だけではなくその周囲も見回してください。

ユーザーインタフェイスについて

5月号の楽譜エディタを見てもわかるように，SX-BASICを使えば独自のコントロールがわりあい簡単に作成できるようになります。楽譜エディタのほかに，独自のコントロールを使用したものとして3月号の付録ディスク中「disk2 sxbasic¥sample」ディレクトリに収められていた「btest.sxb」があります。これはMacintoshやWindowsのようなスクロールバーを実現したものです。

極論すれば，あらゆる種類のコントロールがビットマップアイテムにより表示できますし，ある程度の動きならばSX-BASICのプログラムで対応できます。システムで用意されていなくても，ちょっとしたコントロールならば，SX-BASICで実現できてしまうわけです。

さて，ウィンドウシステムの特徴のひとつに「統一された操作環境」というのがありました。どのアプリケーションでも基礎となる操作方法を統一しておけば，最小限の知識で新しいアプリケーションの使用方法を習得できるというものでした。逆にいえば，このような特徴を発揮できるようにするためにはユーザーインタフェイスはできる限り統一しなければなりません。

ところが，最近，困ったことにシャープ製アプリケーションでも用語の使い方にばらつきが出てきたようです。

　　ロード　開く（オープン）　新規
　　セーブ　保存　新規保存
　　閉じる（クローズ）　終了
　　カット　消去

また，ウィンドウの扱いについても少々変わってきており，たとえば，シャーペンにファイルアイコンを放り込むともうひとつシャーペンのウィンドウが開き，そこにファイルの内容が表示されます。

おそらく，みなMacintoshを真似てわざわざこのような体系にしたのでしょう。

最近では，MacintoshのほうがSX-WINDOWの操作体系に近づいている部分もありますので，Macintoshの操作方法が必ずしも最適というわけではないようです。特別な理由がない限り，1タスク＝1ウィンドウとし，「終了」と「閉じる」を区別したり，「新規」と「開く」を区別したりする必要は感じられないのですが……。

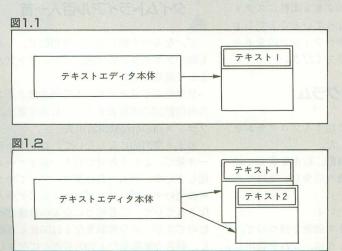
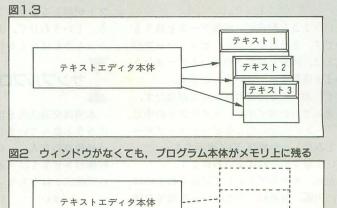
newメソッドの利用について

そろそろコンパイラを用意しようかな，などと構想を練っていますが，いまになって，しまった，と思ったことがあります。

ティブウィンドウを変えた場合も，ウィンドウだけでなくメニューバーの内容も変化する)。そして，メニューバー上から「終了」を選ぶことによりプログラム本体がメモリ上から姿を消します。

なんらかのウィンドウが開かれている状態で「終了」を選択すると，影響はカレントウィンドウに留まらず，すべてのウィンドウが閉じられ，さらにプログラム本体もメモリ上から消えます。

やはり，これでは使いづらいのか，最近はドラッグ＆ドロップという指定方式が採用されました。これはSX-WINDOWのクリーナの扱い方とほぼ同じです。たとえばExcelのファイルを読もうと思ったら，そのファイルアイコンをエクセルのアイコンの上までドラッグするのです（不要になったファイルのファイルアイコンをクリーナのアイコンの上までドラッグするのと同じ動作)。

でもって，最近ではさらにSX-WINDOWライクに，ファイルアイコンを動作中のウィンドウに放り込むことによって，そのプログラムにファイルを読み込ませることもできるようになっているようです（ただし，放り込んだウィンドウに読み込まれるのではなく，新しいウィンドウが開いてそこに読み込まれる)。

現在のところ，SX-BASICで記述されたプログラムをC言語へ変換し，ウィンドウエンジン部分をライブラリとして用意する，という方針を考えていますが，ほとんどのデータは静的に管理されるはずです。SX-BASICは中間コード形式で実行されているので，ほとんどのデータは静的に管理されています。ですからこの部分はコンパイラにとって特に問題にはなりません。しかし，ウィンドウエンジンのアイテム管理部分は動的管理なのです。

ウィンドウデザイナによって配置されたアイテムは，おそらく静的に管理できるのですが，newメソッドにより作られたアイテムはコンパイル時には把握できません。実行されてから初めてウィンドウ上に現れるアイテムです。

やろうと思えば，関数名とそのアドレスとの対応表を用意しておくなどの抜け道はないこともないのですが，余計な実行時間もかかってしまいますし，あまりスマートではありません。

現時点では，どうなるかはわかりませんが，動的管理を匂わせるnewメソッドの存続はあまり可能性が高くありません。なるべく使用は避けてください。

なお，配列アイテムの要素をnewメソッドにより増やす，というのはほとんど静的管理できますので，おそらく存続されるものと思われます。

PICT

画像データの扱いについて，スクリプト形式とビットマップ形式の違いを4月号で説明しました。スクリプト形式とは，

　座標(10,10)から(30,30)に線を引け
　座標(15,15)を頂点として半径10の円を描け
　　　　　　　　：

というような形式で，画像データを扱うものです。それにひきかえ，ビットマップ形式とは，画面に表示されているデータをそのまま保管する，というものです。

で，あっちを立てればこっちが立たず，と迷っていたのですが，スクリプトの中に，

「ハンドルnnの内容をビットマップデータとみなし，座標(X,Y)に表示する」

という命令があればよいわけです。

元来，スクリプトとして記録されるのは，線，円弧，四角形，などといった簡単な図形だけでした。

どのような図形が記録されるのかを調べるには，関数リファレンスで「スクリプトに記録されます」と書かれているかどうかを調べればわかります。たとえば，線を描く命令はGMLineですが，やはり「スクリプトに記録されます」と書かれています。プロットデータを表示するのはGMPlotImgですが，残念ながら，ここには「スクリプトに記録されます」とは書かれていません。もし，この命令がスクリプトに記録されるなら，3月号のジレンマは解決するわけです。すべてスクリプトとして扱い，その中でプロットデータを扱うという感じになります。

また，Easydrawの発表後，クリップボードでドローデータ（前述のスクリプト形式と専用形式の2本立て）が扱えることが判明しました。SX-BASICがクリップボードに対応した暁には，文字列データだけでなく画像データも扱いたいものです。クリップボードに対応するとペースト先でそのデータがどうのような扱いを受けるかはわかりません。半分に縮小されるかもしれませんし，10倍に拡大されるかもしれません。図形データはプロットデータでなく，なるべく，スクリプト形式でやりとりしたいものです。

SX-WINDOWを解析した人の話によると，ver.3.0になってスクリプトの扱いが大幅に拡張されたそうです。ベジェ曲線やスプライン曲線を扱うらしい命令がある，とのことです。残念ながら，プロットデータの表示はなさそうでしたが。

また，スクリプトの命令をユーザーが自由に拡張したり，書き換えたりできるようになっているかもしれない，とのことですので，ない命令は拡張してしまうというのもひとつの手かもしれません。

解析を始めよう（始めてもらおう），と思った矢先のver.3.1の発表でしたが，おそらくver.3.1では，ver.3.0よりも柔軟にスクリプトが扱えるようになっていることでしょう。というわけで，グラフィックのサポートはもうしばらくお待ちください。

サンプルプログラム

本当は交通法規を出題するプログラムを作ろうと思っていました。

「無免許の友人に，自動二輪車を貸して運転練習させても，人通りの少ない道路であれば違法とはならない」
　　　はい　　　いいえ

「覚醒剤は眠気を覚ます効能を持つので，車を長時間運転するときは，これを使用して運転するとよい」
　　　はい　　　いいえ

みたいなやつ。

そうすれば，免許取得中という丹氏がSX-BASICに興味を持ち，いろいろとアドバイスをもらえるかもしません。ワークステーションでのトレンド，オブジェクト指向などに関するアカデミックな示唆が得られれば，現在のSX-BASICにとって，これほどありがたいものはありません。

で，

石「教習所の授業って，耐えられます？」
丹「いやぁ，私は大学時代に人間を鍛えてたから」
石「でも，あの授業であの試験に受かれというのは，ちょっと酷ではないですか」
丹「だから，私は人間ができてるから。それに，本番そっくりの問題集もあるし」
石(身を乗り出して)「パソコン上で模擬練習できたらいいと思いません？」
（そこへ突然，坪井氏が現れる）
坪「ところで，丹さん，車，買ったのですか」
丹「いやぁ，駐車場とかの関係で，なかなか買えなくて。早く車を買わないと運転の仕方を忘れちゃいますよ」
坪「どのくらい，乗ってないんですか？」
丹「免許とってから，まったく運転してませんから，もう1カ月くらいかな」
石「えっ，丹さん，もう免許とったんですか」
丹&坪「は？」

というわけで，「タイムトライアル道交法」は没ということになりました。

SX-BASICがマルチフォントテキストに対応した際にでも，復活するかもしれません（交通標識とかもテキスト中に張り込めるようになるしね）。

タイムトライアル百人一首

で，なるべく楽しめて，実用的で，しかも短いプログラムで実現できて，サンプルとして適切なもの。

世界史の年号クイズ，県庁所在地クイズ，古典助動詞の活用表クイズ，日本国憲法，フランス語のêtre動詞活用表……。

あまり専門的にならないで，ほどよいデータ量で，というわけで百人一首をゲーム化しました。百人一首は某ワープロでマクロ機能のサンプルプログラムとしてゲーム化されていて，二番煎じになるのは嫌だったのですが，古今和歌集だと1100首もあるし，新古今和歌集だと2000首近くになるし，かといって，サラダ記念日なんて暗記して

もあんまり自慢にならないし，だいいち，引用許可が下りるわけもないしというわけで，結局，百人一首になりました。

クイズのデータとして百人一首を用いるだけですから，このデータを変えれば，さまざまなゲームへと変形するわけです。

(て)氏のショートプロぱーてぃでも，SX-BASICを用いた投稿を扱ってくれるようなので，面白いものができたら送ってみるのもよいかもしれません（ただし，データだけ変えて送っても，採用される確率は低いでしょう)。

遊び方

ウィンドウ上に百人一首の上の句が表示されますので，なるべく早く下の句を選んで，そのボタンを押してください。間違えると正解がダイアログで表示されますので，これを踏まえて正しい解答を選択してください。5首終わるとそれまでの経過タイムがダイアログ上に表示されます。

ゲームの終了などの処理は特別なにも行っていませんので，適当に遊んで飽きたらウィンドウエンジンのクローズボタン（ウィンドウ右上端にあるボタン）を押して終了します。

プログラムの入力

ウィンドウデザイナを用いて写真1のようなウィンドウを設計してください，といいたいところですが，リスト1をそのままエディタから打ち込んだほうが早いでしょう。

ウィンドウデザイナは，自分で試行錯誤しながらアイテムを配置するには効率的なのですが，打ち出されたリストと同じものを作成する際は，あまり効率的に作業を行えません。

とりあえず，エディタからリスト1を打ち込み，実行し，気にいらない箇所があればウィンドウデザイナを使って修正するのがいちばん効率的な方法かもしれません。

例によって，▼マークのついているのが，ウィンドウエンジン上のアイテムに関するデータです。残りが，X-BASICとほとんど同じコードです。

「▼」は，JISコードの0x2227にあります。ASK3だと「さんかく」を漢字変換すると候補の中にあるはずです。

ひととおり入力が終わったらファイルに保存し，実行してみてください。なにもエラーが起こらなければ，

写真1　このようにウィンドウを作る

写真2　実行例

print kaminoku(98)；simonoku(98)

を実行させてください。kaminokuは上の句が収められた文字列配列，simonokuは下の句が収められた文字列配列です。入力ミスによりひとつの要素でも抜けていたり，余分なものがあったりすると上の句と下の句が正しく対応しません。

今回のプログラムでは99番目には後徳大寺左大臣の，

ほととぎす鳴きつるかたをながむれば
　ただ有明の月ぞのこれる

という句が入っていますので，このとおりに表示されれば上の句と下の句の対応は正しくなっているものと思われます（断定できるわけではないが，ほぼ正しい)。

下の句が，

　われても末にあはむとぞ思ふ

だった場合は，どこか1箇所入力もれがあります。逆に，

　いづくもおなじ秋の夕ぐれ

だった場合には，1箇所余計なデータが混じっています。

上の句も同様にチェックし，入力ミスがある場合は修正してください。

プログラムの説明

今回のプログラムは約300行もありますが，プログラムとしての正味は100行もありません。残りは，ほとんどが百人一首のデータです（上の句が100行，下の句が100行)。

というわけで，206行目まではX-BASICとなんら変わるところのない，変数の宣言，配列変数の初期化です。

209行目から215行目の「▼」で始まる行はSX-BASICに特有のデータです。このデータによって，ウィンドウエンジン上に各種アイテムが配置されるようになります。データのフォーマットについては先月号をご覧ください。

プログラム中，使用している変数は以下のとおりです。

r：100ある問題のうち，どれを出題するか。r=0で0番目の，

淡路島かよふ千鳥のなく声に
　幾夜寝覚めぬ須磨の関守

を表示する。

ans:正解を何番目の選択肢に表示するか。アイテムStnBtn1［ans］がクリックされると正解時の処理を行う。

trial：何度目の問題か。trial=5でゲーム終了。

starttime：ゲームを始めた時間。単位はシステム時間（後述)。

残りのプログラムの構造は，X-BASICとほとんど同じですので特に解説はいらないと思いますが，gettime()関数だけはA line関数を使ったちょうどよいサンプルですので，少し説明しておきましょう。

gettime()関数の実体は，

　return(A_line(&ha0af))

だけです(268行目)。A_line(&ha0af）という関数によって得られた値を返すだけの働きです。プログラムが読みにくくなってしまうのでやってませんが，リスト1中のすべてのgettime()関数は，A_line(&ha0af)と置き換えても，動作は同じです。

で，参考文献2を調べると，

$A0AF EMSysTime

現在のシステム時間（1/100）を返す。

引数

なし

返り値

D0.L　システム時間

と，あります。

システム時間とは，SX-WINDOWが起動されてからの時間を1/100秒単位で表す時間です。SX-BASICにも，似たような関数としてtime$がありますが，値を「時間：分：秒」に分けた文字列として返しますので，減算ができません。ゲームを始めてから終了するまでにかかった時間を求めるには，

n時間　→　60×n分
m分　→　60×m秒

のような変換を行い，すべてを同じ単位に変換しなければなりません。

SX-BASICでは，システム時間を返す機能はありませんでした。

これに対し，システム時間は単位が1/100秒ですから，終了時のシステム時間から，スタート時のシステム時間を引いてやれば，ゲームにかかった時間が得られるわけです。

こうして，得られた値を100で割れば，「秒」になりますので，これを264行で表示しています。

SX-BASICには，最低限の命令しか備わっていません。一応，X-BASICが持っている命令は可能な限り揃えましたが，SX-WINDOWの機能を網羅するには，とても足りません。

SX-BASICにはない機能だけれども，どうしても使いたい場合には，このA_line関数を用いることによって，ある程度のことはできるようになります。

参考文献

1) 伊藤秀文「小倉百人一首のしおり」河出興産

2) 吉沢正敏「SX-WINDOWプログラミング」ソフトバンク

祝アクセラレータ始動！ 投稿歓迎！

そんなわけで，私がとうとうアクセラレータを動かしてしまった石上です。いやー，動くときにはあっさり動くもんです。いま回路図などをまとめているので，発表は来月号まで待っててね。

さて，もうひとつうれしいことといえば，(で)氏の担当する「ショートプロぱーてぃ」でもSX-BASICを使ったプログラムを扱ってくれるようになったことです。6月号でもさっそく市川氏による「GAME.SXB」が掲載されていましたね。

さらに，ショートプログラムというわけではないのですが，SX-BASICを使ったかなりの大きさのプログラム投稿も編集部に届いているようです。ちょっと大きすぎて，発表が付録ディスクまでおあずけになってしまうのですが，これは本当に残念です（プログラム自体よりもグラフィックリソースがでかい）。

やっぱ，うれしいですよね。

はっきりいって，SX-BASICを5月号の段階まで盛り上げてくれたのはデバッグ担当の中野氏です。ひと癖もふた癖もあるバグ出し用のプログラムにはずいぶん驚かされました。

初めて楽譜エディタを見たときには，まさかそれがSX-BASICで書かれたプログラムだとは思いませんでした（わはは）。

このように，反応があるというのは気持ちがよいものなので，私も(で)氏を気持ちよくしてあげましょう。ということで，ここで「ショートプロぱーてぃ」への逆乱入を行いたいと思います。

5月号では「使えるSXアプリなのだ！」ということで仁泉大輔氏によるOPMSX.Xというプログラムが掲載されていました。これはSX-WINDOW上で打ち込まれたテキストをMMLとみなし，その場で再生させてしまおうというプログラムです（C言語で184行）。

これをSX-BASIC版にしたのが右下のリストです。OPMDRV3.Xなり，ZMUSIC.Xなり，なんらかのOPMドライバを組み込んだうえでSX-WINDOW上から（SX-BASIC上から）起動してください。

字下げや空白行があるので一概には比較できないのですが，OPMSX.Xに比べても入力すべきコード量が1割以下になっているのがわかると思います。

簡単に説明しておくと，

1行目　ウィンドウの大きさ（6月号参照）

2行目　テキストアイテムのデータ

3行目以降　リターンキーが押された場合の処理

で，OPMの初期化データ＋与えられた文字列をシステム予約ファイル名「OPM」にコピーしてます。

＊　＊　＊

ということで，SX-WINDOWのプログラムが簡単に作れるSX-BASICをこれからもよろしくお願いします。

るん♪

```
 10 ▼Window size (300,100),0,0,OPMSX
 20 ▼1,Text1 (12,8,288,88),0,0,1,0,3,1,0,1,
 30 func Text1?_KeyIn(text;str)
 40 int fn
 50 fn=fopen("opm")
 60 fwrites("(i)(m1,1000)(a1,1)",fn)
 70 fwrites("(t1)"+text+"(p)",fn)
 80 fclose(fn)
 90 endfunc
100 func Text1_Click()
110 endfunc
```

リスト

```
 1: ▼Window Size (202,247),0,0,T.T.百人一首
 2: int i,r,ans,trial,starttime
 3: dim str kaminoku(100) = {
 4: "淡路島かよふ千鳥のなく声に",
 5: "あはれともいふべき人は思ほえで",
 6: "嵐ふく三室の山のもみち葉は",
 7: "あらざらむこの世のほかの思ひ出に",
 8: "秋の田のかりほの庵の苫をあらみ",
 9: "秋風にたなびく雲のたえ間より",
10: "天の原ふりさけ見れば春日なる",
11: "天つ風雲のかよひち吹きとぢよ",
12: "ありま山猪名の笠原風吹けば",
13: "有り明けのつれなく見えし別れより",
14: "あひみての後の心にくらぶれば",
15: "あけぬれば暮るるものとは知りながら",
16: "足引の山鳥の尾のしだり尾の",
17: "浅芽生の小野の篠原しのぶれど",
18: "朝ぼらけ宇治の川霧たえだえに",
19: "朝ぼらけ有明の月とみるまでに",
20: "名にしおはば逢坂山のさねかづら",
21: "難波江の芦のかりねの一夜ゆゑ",
22: "難波潟みじかき芦のふしの間も",
23: "夏の夜はまだ宵ながら明けぬるを",
24: "嘆きつつひとりぬる夜の明くる間は",
25: "嘆けとて月やは物を思はする",
26: "長からむ心も知らず黒髪の",
27: "ながらへばまたこの頃やしのばれむ",
28: "大江山いく野の道の遠ければ",
29: "おほけなくうき世の民におほふかな",
30: "逢ふことのたえてしなくばなかなかに",
31: "奥山にもみぢふみわけなく鹿の",
32: "思ひわびさても命はあるものを",
33: "音にきく高師の浜のあだ浪は",
34: "小倉山みねのもみぢば心あらば",
35: "わが庵は都のたつみしかぞすむ",
36: "わが袖は潮ひにみえぬ沖の石の",
37: "忘らるる身をば思はずちかひてし",
38: "忘れじの行末まではかたければ",
39: "わびぬれば今はたおなじ難波なる",
40: "わたの原こぎいでてみれば久方の",
41: "わたの原八十島かけてこぎいでぬと",
42: "高砂のをのへの桜さきにけり",
43: "立ち別れいなばの山の峰に生ふる",
44: "田子の浦うち出てみれば白妙の",
45: "玉のをよ絶えなば絶えねながらへば",
46: "滝の音はたえて久しくなりぬれど",
47: "誰をかもしる人にせむ高砂の",
48: "心あてに折らばや折らむ初霜の",
49: "心にもあらでうき世のながらへば",
50: "これやこの行くも帰るもわかれては",
51: "このたびはぬさもとりあへず手向山",
52: "こぬ人を松帆の浦の夕なぎに",
53: "恋すてふわが名はまだき立ちにけり",
54: "みかの原わきて流るるいづみ川",
55: "みかきもり衛士のたく火の夜はもえ",
56: "陸奥のしのぶもちずり誰ゆゑに",
57: "みよし野の山の秋風さよふけて",
58: "見せばやな雄島のあまの袖だにも",
59: "春すぎて夏来にけらし白妙の",
60: "春の夜の夢ばかりなる手枕に",
61: "花の色はうつりにけりないたづらに",
62: "花さそふ嵐の庭の雪ならで",
63: "やすらはで寝なましものを小夜ふけて",
64: "山里は冬ぞさびしさまさりける",
65: "山川に風のかけたるしがらみは",
66: "八重むぐらしげれる宿のさびしさに",
67: "夜もすがら物思ふころは明けやらで",
68: "世の中よ道こそなけれ思ひ入る",
69: "世の中は常にもがもな渚こぐ",
70: "夜をこめて鳥のそらねははかるとも",
71: "かささぎの渡せる橋におく霜の",
72: "かくとだにえやはいぶきのさしも草",
73: "風そよぐならの小川の夕ぐれは",
74: "風をいたみ岩うつ波のおのれのみ",
75: "今こむといひしばかりに長月の",
76: "今はただ思ひ絶えなむとばかりを",
77: "いにしへの奈良の都の八重桜",
78: "ちはやぶる神代もきかず竜田川",
```

```
 79: "ちぎりきなかたみに袖をしぼりつつ",
 80: "ちぎりおきしさせもが露をいのちにて",
 81: "ひさかたのひかりのどけき春の日に",
 82: "人はいさ心もしらずふるさとは",
 83: "人もをし人もうらめしあぢきなく",
 84: "君がため春の野にいでて若菜つむ",
 85: "君がため惜しからざりし命さへ",
 86: "きりぎりす鳴くや霜夜のさむしろに",
 87: "憂かりける人を初瀬の山嵐",
 88: "うらみわびほさぬ袖だにあるものを",
 89: "月みればちぢに物こそかなしけれ",
 90: "つくばねの峰よりおつるみなの川",
 91: "忍ぶれど色に出でにけりわが恋は",
 92: "白露に風のふきしく秋の野は",
 93: "もろともにあはれと思へ山桜",
 94: "ももしきやふるき軒端のしのぶにも",
 95: "夕されば門田の稲葉おとづれて",
 96: "由良のとをわたる舟人かぢをたえ",
 97: "村雨の露もまだひぬまきの葉に",
 98: "住の江の岸による波よるさへや",
 99: "めぐりあひて見しやそれともわかぬまに",
100: "ふくからに秋の草木のしをるれば",
101: "さびしさに宿をたちいでてながむれば",
102: "ほととぎす鳴きつるかたをながむれば",
103: "瀬を早み岩にせかるる滝川の"
104: }
105: dim str simonoku(100) = {
106: "幾夜寝覚めぬ須磨の関守",
107: "身のいたずらになりぬべきかな",
108: "竜田の川の錦なりけり",
109: "今ひとたびの逢ふこともがな",
110: "わが衣手は露にぬれつつ",
111: "もれいずる月の影のさやけさ",
112: "三笠の山にいでし月かも",
113: "をとめの姿しばしとどめむ",
114: "いでそよ人を忘れやはする",
115: "あかつきばかりうきものはなし",
116: "昔はものを思はざりけり",
117: "なほ恨めしき朝ぼらけかな",
118: "なかながし夜を独りかも寝む",
119: "あまりてなどか人の恋しき",
120: "あらはれ渡るせぜの網代木",
121: "吉野の里にふれる白雪",
122: "人に知られでくるよしもがな",
123: "身をつくしてや恋わたるべき",
124: "逢はでこの世をすぐしてよとや",
125: "雲のいづこに月やどるらむ",
126: "いかにも久しきものとかは知る",
127: "かこち顔なるわが涙かな",
128: "乱れてけさはものをこそ思へ",
129: "うしとみし世ぞいまは恋しき",
130: "まだふみも見ず天の橋立",
131: "わがたつ 杣 に黒染の袖",
132: "人をも身をも恨みざらまし",
133: "声聞くときぞ秋はかなしき",
134: "うきに絶へぬは涙なりけり",
135: "かけじゃ袖のぬれもこそすれ",
136: "今ひとたびのみゆきまたなむ",
137: "世をうち山と人はいふなり",
138: "人こそしらねかわくまもなし",
139: "人の命の惜しくもあるかな",
140: "今日かぎりの命ともがな",
141: "みをつくうしても逢はむとぞ思ふ",
142: "雲井にまがふ沖つ白波",
143: "人にはつげよあまのつり舟",
144: "とやまの霞たたづもあらなむ",
145: "まつとしきかば今帰りこむ",
146: "富士のたかねに雪は降りつつ",
147: "忍ぶることの弱りもぞする",
148: "名こそ流れてなほ聞えけれ",
149: "松も昔の友ならなくに",
150: "おきまどはせる白菊の花",
151: "恋しかるべき夜半の月かな",
152: "しるもしらぬも逢坂の関",
153: "紅葉の錦神のみにまに",
154: "焼くやもしほの身もこがれつつ",
155: "人しれずこそ思ひそめしか",
156: "いつみきとてか恋しかるらむ",
157: "昼は消えつつ物をこそ思へ",
158: "みだれたそめにし我ならなくに",
159: "ふるさと寒く衣うつなり",
160: "ぬれにぞぬれし色はかはらず",
161: "衣ほすてふ天のかぐ山",
162: "かひなく立たむ名こそ惜しけれ",
163: "わが身世にふるながめせしまに",
164: "ふりゆくものはわが身なりけり",
165: "かたぶくまでの月をみしかな",
166: "人めも草もかれぬと思へば",
167: "ながれもあへぬ紅葉なりけり",
168: "人こそ見えね秋は来にけり",
169: " 閨 のひまさへつれなかりけり",
170: "山の奥にも鹿ぞ鳴くなる",
171: "あまの小舟のつなでかなしも",
172: "よに逢坂の関はゆるさじ",
173: "白きを見れば夜ぞふけにける",
174: "さしも知らじな燃ゆる思ひを",
175: "みそぎぞ夏のしるしなりける",
176: "くだけて物を思ふころかな",
177: "ありあけの月を待ち出でつるかな",
178: "人づてならでいふよしもがな",
179: "けふ九重ににほひぬるかな",
180: "からくれなゐに水くくるとは",
181: "末のまつ山波こさじとは",
182: "あはれことしの秋もいぬめり",
183: "しづ心なく花の散るらむ",
184: "花ぞむかしの香ににほいける",
185: "世を思ふゆゑに物思ふ身は",
186: "わが衣手に雪は降りつつ",
187: "長くもがなと思ひけるかな",
188: "衣かたしきひとりかも寝む",
189: "はげしかれとは祈らむものを",
190: "恋にくちなむ名こそ惜しけれ",
191: "わが身ひとつの秋にはあらねど",
192: "恋ぞつもりて淵となりぬる",
193: "物や思ふと人のとふまで",
194: "つらぬきとめぬ玉ぞ散りける",
195: "花よりほかにしる人もなし",
196: "なほあまりある昔なりけり",
197: "芦のまろやに秋風ぞ吹く",
198: "ゆくへも知らぬ恋のみちかな",
199: "霧たちのぼる秋の夕ぐれ",
200: "夢のかよひ路人めよくらむ",
201: "雲がくれにし夜半の月かな",
202: "むべ山風を嵐といふらむ",
203: "いづくもおなじ秋の夕ぐれ",
204: "ただ有明けの月ぞのこれる",
205: "われても末にあはむとぞ思ふ"
206: }
207: GameStart()
208: end
209: ▼1,Text1 (12,8,192,36),0,0,0,0,3,0,0,1,上の句
210: ▼1,Text2 (12,44,192,76),0,0,0,1,3,0,1,1,下の句を選んで下さい
211: ▼3,StnBtn1 (12,88,196,108),1,0,
212: ▼3,StnBtn1 (12,120,196,140),1,1,
213: ▼3,StnBtn1 (12,152,196,172),1,2,
214: ▼3,StnBtn1 (12,184,196,204),1,3,
215: ▼3,StnBtn1 (12,216,196,236),1,4,
216: func StnBtn1_Click(i;int)
217:   if(ans == i ) then {
218:     trial = trial + 1
219:     if(trial == 5) then GameEnd() else drawQuestion()
220:   } else {
221:     alart(&hff01, kaminoku(r)+chr$(13)+simonoku(r))
222:   }
223: endfunc
224: /* 問題(上の句&5つの下の句)を表示
225: func drawQuestion()
226:   r = rand() mod 100
227:   Text1.caption = kaminoku(r)
228:   /* 正解をどこに表示するか
229:   ans = rand() mod 5
230:   drawChoice()
231: endfunc
232: /* 選択肢を表示
233: func drawChoice()
234: int i, r1
235: for i=0 to 4
236:   if(ans == i) then {
237:     StnBtn1[i].caption = simonoku(r)
238:   } else {
239:     repeat {
240:       r1 = rand() mod 100
241:       if(r == r1) then continue /* 正解を表示しないように
242:       StnBtn1[i].caption = simonoku(r1)
243:     } until(1)
244:   }
245: next
246: endfunc
247: /* ゲームスタート
248: func GameStart()
249:   int i
250:   for i=0 to 4
251:     StnBtn1[i].enable = 1
252:   next
253:   trial = 0
254:   starttime = gettime()
255:   drawQuestion()
256: endfunc
257: /* ゲーム終了
258: func GameEnd()
259:   int i, time, ret
260:   for i=0 to 4
261:     StnBtn1[i].enable = 0
262:   next
263:   time = (gettime() - starttime) / 100
264:   alart(1, "所要時間は"+itoa(time)+"でした")
265: endfunc
266: /* システム時間を得る
267: func gettime()
268:   return(A_line(&ha0af))
269: endfunc
270: func Text1_Click()
271: endfunc
272: func Text2_Click()
273: endfunc
```

仮想ドライバの開発実験PART4.

対PC-9801接続への挑戦

電机本舗 由井 清人 Yui Kiyoto

X68000同士を接続した基本実験が終わったところで，今回は他機種との接続を目指して実験を行います。ターゲットはPC-9801シリーズです。それではX68000からPC-9801のディスクを読み取ってみましょう。

今回は前回に引き続き外部のX68000に仮想ドライブディスクとして，ほかのパソコンのディスクを利用します。

前回はX68000を2台接続し，2台目のほうを従機として利用できるようにしました。従機のフロッピーディスク，ハードディスクを主機のほうからもドライブとして認識できるというものです。

今回は，この従機側をさらに押し進めてPC-9801を接続できるようにしてみました。そして，PC-9801のディスクをX68000より読み書きできるか実験してみます。接続は前回同様RS-232Cコネクタを利用します。

MS-DOS機への接続

Human68kシステムはMS-DOSのファイル管理システムをお手本にして作られています。ですから，これまで作ってきた仮想ドライブシステムはそのままPC-9801でも動作する可能性が濃厚です。ここでは，第1回目の実験報告としてかなり興味深い（動作する）データを採取できましたので，ご案内します。

●使用したDOS環境

今回は，PC-9801系で実験してみました。厳密な環境を次に示します。

1) 機種　PC-286LH10（エプソンの98互換機）
2) CPU＝V30；クロック＝10MHz,内蔵HDD＝10MB1基，FDD=2HD1基，メモリ640Kバイト
3) 開発環境　Quick-C ver.2
4) MS-DOS ver.3.1

利用したマシン環境はかなりの老朽機です。基本的にはDOSのバージョンの影響はあっても機種の影響はありません（まったくないとはいいませんが）。

PC-9801を選択した理由

今回はMS-DOSマシンといってもPC-9801だけを考慮しました。いまはやりのDOS/V（IBM PC互換機）は無視しました。これは，開発の難易度によります。

PC-9801系はRS-232Cの制御機能をBIOS（Basic Input Output System＝基本入出力システム）できちんと搭載しています。BIOSとは，X68000でいうところのIOCSに相当するもので，通常はROMに格納されています。性能の良し悪しは別として，機能をきちんとアプリケーションに対して提供しています。

対してIBM PC系では，BIOSにあることはあるのですが実用に供しません。1バイト程度の入出力であれば問題ないのですが，2バイト以上のデータが入ってくるとデータの取りこぼしが発生します。これは受信バッファを持っていないためです。2バイト目のデータが1バイト目に上書きされ，消失します。これにより，普通にBIOSを使用しているかぎりは，ほぼ間違いなくプログラムは心不全を起こします。

今回は対MS-DOS接続に焦点を絞って実験をしました。このようなわけで，比較的素直に接続できるであろうPC-9801を選択した次第です。

MS-DOS接続の勘どころ

詳細は後述しますが，PC-9801の接続そのものは比較的素直に行えました。これは，X68000上で作った従機のプログラム「R.X」（ソースコード名はR.C）をスムーズに移植できたということです。

移植に際し問題となったのは，Cコンパイラの相違部分，およびCPUの特性に関連するところでした。

●i8086 VS MC68000

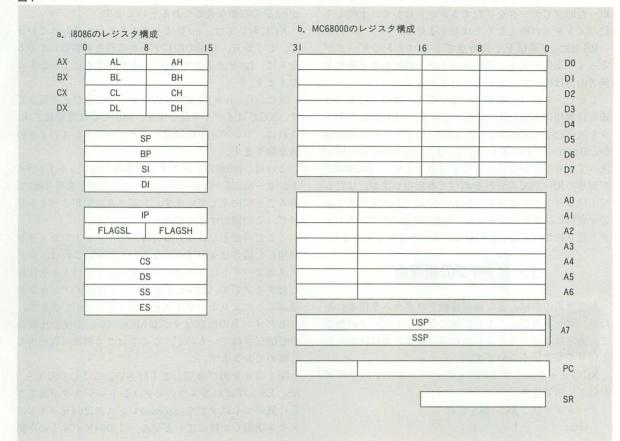
i8086とMC68000というのはかなり相違点のある石です。まず，図1に両者のレジスタ（CPU内部の作業メモリ）構成を示します。

このあたりからくる相違がまず視覚的な点でしょう。特に，両者のレジスタが，i8086が16ビット長で，MC68000が32ビット長であることに気をつけてください。

さらに，各レジスタの上にビット並びの番号を振っておきました。これは概念的なものですが，これから説明することをわかりやすくするために振りました。注意して見てください。i8086とMC68000でビットの番号の並び方が正反対であることに気がつくはずです。

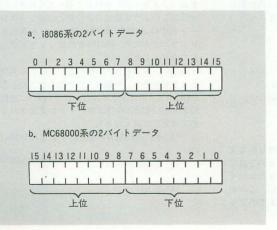
すなわち，i8086が0ビット目から始まり15ビット目で終わる昇順であること，対するMC68000が31ビット目から始まり0ビット目で終わる降順であることです。

この違いこそが実は双方間の互換性，移植性を決定的に損なっているのです。また，この違いは，設計思想，理念の相違といった高尚なものではなく，単に「左巻き」であるか「右巻き」であるかといったものであると筆者

図1

は睨んでいます。偶然，ともに「右巻き」であればよかったのですが，結果は1/2の確率で相反する方式に傾いたのではないでしょうか。

この並びがなぜ問題となるのか。これは，たとえば16ビットのデータをメモリ空間に横に平べったく展開してみれば一目瞭然です。

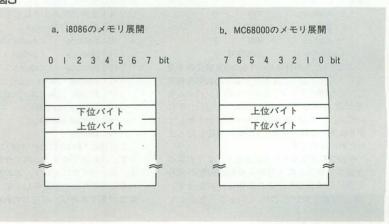
図2のaにインテル系（i8086）を，bにモトローラ系（MC68000）を表示します。

パソコンでは通常はメモリ空間をバイト単位で区切って活用しています。ですから，この16ビットのデータをメモリ空間に展開すると図3のようになります。aがインテル系（i8086）でbがモトローラ系（MC68000）です。

このときに，16ビット，すなわち2バイトの上位バイトと下位バイトが両者で入れ替わっていることに気がつくはずです。この時点で両者は決定的にデータの互換性を失っているのです。

普通はメモリの利用方法が違うというだけであまり問題にはなりません。しかし，フロッピーディスクやハードディスクに書き込まれた2バイト，4バイト長のデータを「逆巻き」のCPUが読み込んだときに悲劇が生じます。上位バイトと下位バイトが逆転してCPU内に取り込まれ，でたらめな値と化します。

この弊害は今回の仮想ドライブシステムにも濃厚に表れています。主機と従機は常時データを交信しています。このときに，特に操作を施さずデータを従機（PC-9801）

図2

図3

側に送信しています。ですから，PC-9801はデータを受け取った時点でデータを「たすきがけ」にして，上位と下位バイトを入れ換えなくてはなりません。

RS-232Cで通信をした時点でも1バイトのデータがそっくり上位ビットと下位ビットが入れ換わる可能性を読者の方は抱いているかもしれません。

しかし，それは杞憂です。なぜなら，通信そのものは通常は専用の通信LSIが担当しており，CPUはただデータを通信LSIへ送ればよく，実際のビット列の送出は通信LSIが行ってくれるからです。また，送出順序は仕様が決っており並びが変化することはありません（執筆時点で簡単にRS-232Cの資料を調べてみたのですが，はたして上位ビットから送出するのか下位ビットから送出するのか資料には見あたりませんでした。完全にブラックボックス化されていて知る必要がないということでしょうか）。

データタイプの相違点

異機種間の移植，また通信接続プログラムを作るにあたり，データ並びとともに問題になるのがデータの型です。通常はこの2つを片付ければ，ほぼ作業は終わったといってもよいくらいです。

XCのデータタイプとMS-Cのデータタイプのサイズを以下に示します。

データタイプ	XC	MS-C系
char	1	1
int	4	2
long int	4	4
char＊	4	2(可変)　　バイト

ここで，int型とchar＊型に注意してください。int型は，もはやC言語の古典的文法書となったカーニハンとリッチーの「C言語入門」に従うならば，C言語の稼働しているマシンにおいて一番自然な，つまり扱いやすいデータ長であると記述されています。

先の図1を見てください。i8086の各レジスタは2バイトであることから，MS-DOSマシンにおいては2バイトがint型の自然な長さであるといえます。

XCにおいては，int型は4バイトとして扱われています。ですから，移植の際にデータの長さが影響してくるならば，intを使用せずlong intを使用すればよいことになります。

次に，「char＊」を見てください。ここはひどく厄介です。XCでは4バイト長ですが，MS-Cでは可変長です。これは，コンパイラの設定により2バイトか4バイトの値を取ります。

これは，i8086のレジスタが基本的に16ビット（2バイト，0～65535＝64Kバイトのメモリ空間を表現可能）であることに由来しています。レジスタの幅が2バイトなので，この値がサイズになりうるのです。

ここで可変となっているのは，セグメントレジスタを利用した場合は4バイトになるからです。これは，レジスタの2バイトにセグメント開始を示す2バイトを付加したタイプです。MS-Cではコンパイルのオプションを選ぶことによりこのサイズを利用することができます。

セグメントの概念はすでにMC68000とi8086の比較の中で喧伝されているので，ここではごく簡単に説明するに留めておきます。

図1のaを再び参照してください。ここにDS，CS，SS，ESと呼ばれる4つのセグメントレジスタがあることに気がつくはずです。i8086のレジスタは64Kバイトのメモリ空間しか持っていません。この64Kバイトの空間をi8086はセグメントと呼んでいます。

i8086は，このセグメントを最大4本まで持つことができ，それぞれを1Mバイトの空間にちらかすことができます。

すなわち，各セグメントの開始アドレスを示す2バイトの位置情報として4つのセグメントレジスタ，DS(Data Segment:データ領域指定専用)，CS(Code Segment:プログラムコード領域指定専用)，SS(Stack Segment:スタックデータ領域指定専用)，ES(Extra Seg

移植のテクニック1

今回のプログラムは前回作ったものを改造して作りました。リスト2の先頭で次のような宣言をしています。

```
1：#define        X68        0
2：#define        PC98       1
3：#define        IBM        2
4：#define        SYSTEM     1
```

ここでは，SYSTEMというdefine定義定数を用意しました。ここに0から2までの数字を指定し，その値によりソースコードのコンパイル状況が変わるようにしました。0のときにX68000，1でPC-9801用のソースコードとして働きます。2は将来IBM互換機に対応したときのための予約です。

今回プログラムを作るにあたり，まず前回の従機プログラムのうちからX68000固有の部分を選り分けました。この部分は，削除せず，define定義定数SYSTEMが0のときに有効となるように変更しました。さらに，今回新たに作った部分，PC-9801固有のところはSYSTEMが1のときに有効となるようにしました。具体的にはリスト2中で次のようにしています。

```
   9：#if SYSTEM==X68
+ 10：#include    <doslib.h>
  11：#include    <stdio.h>
  12：#include    <time.h>
  13：#define   blk_in_x(a,b) blkin(a,b)
  14：#define   blk_out_x(a,b) blk_out(a,b)
  15：#define XCHG2(a)
  16：#define XCHG4(a)
  17：#endif
```

9行目の「#if SYSTEM==X68」は特殊な判定文です。これはプログラム内で作用するのではなく，コンパイラにおいて，この部分をコンパイルするかどうかの判定をします。もし，この判定文が真であれば，17行目の#endifまでの文を有効としてコンパイルします。偽であれば17行目までが丸々存在しないものとして扱います。まあ，コメント扱いされると思ってください。今回はこのような手法によりひとつのソースコードの中にX68000の機種依存コードを混在させつつ，さらにPC-9801特有のコードを混ぜ込んで作りました。

同様に，次の行はPC-9801のときにのみ有効となる部分です。今回は，この部分が有効となりQuick-C上でコンパイルリンクされます（定数SYSTEMとPC98は今回は等しく1に定義されるので判定文は真となる）。

```
19：#if   SYSTEM==PC98  ||  SYSTEM==IBM
20：#pragma pack(1)
     ：
84：#endif
```

ment:その他領域指定専用)を持ちます。

「char＊」に代表されるポインタ型を利用した場合，このセグメントレジスタの幅が加算され，結果として4バイト長になるのです。

今回の仮想ドライブシステムは，スモールモデルと呼ばれるセグメントレジスタを使用しないコンパイルオプションを利用しているので，ポインタは2バイトとしてプログラム中では扱われます。当然，構造体はXCとは同じコーディングをすると大きさなどが異なります。ですから，きちんと両者の長さをあわせてあげないと転送したデータがずれていくので注意が必要です。

実際のプログラミング

今回は従機側のみ作る予定でしたが，主機側もある程度手直しが必要でした。

X68000用の主機側のプログラム「D1.C」をリスト1と差し替えてください。コンパイル方法は5月号を参照してください。

さて，肝心の従機側のプログラムです。次に構成を示します。

リスト2　R.C　　C言語による内部処理プログラム
リスト3　D3.C　通信の入出力ルーチン

このプログラムは，おそらくDOS系のどのC処理系でも動作すると思います。コンパイルするときにはスモールモデルを指定してください。および，構造体のパックオプション(MS-Cでのコンパイルオプションは/Zp)を指定することです。パックオプションはソースコード中にじかに埋め込んであるので，Quick-C上であれば問題ありません。次のようにR.Cの20行目にパックを指定する#pragma pack(1)文が入っているからです。この行はほかのC処理系ではエラーになるかもしれません。その場合はコンパイルオプションでパックを指定すれば同じことになります。

```
20： #pragma pack(1)
```

詳しいプログラミングに関しては，コラムへ切り分けておきました。こちらを参照してください。

うまくコンパイルリンクに成功すれば，「R.EXE」という実行形式のファイルができているはずです。

使用方法

まず，PC-9801の環境設定の確認を行います。今回のシステムはBIOSによりRS-232Cを制御しています。ですから，MS-DOS上で用意しているRS-232Cは使用してはいけません。PC-9801には，本来のBIOSによるRS-232Cの制御方法とは別にMS-DOS付属のRS-232Cの制御システムがあります。これらは別のものと考えてください。ここではMS-DOS付属のほうは使用しません。もし，環境設定ファイルCONFIG.SYSの中に次の文が入っていれば削除してしまってください。

```
DEVICE=RSDRV.SYS
```

これは，MS-DOSのRS-232C制御システムだからです。

これで，いよいよ仮想ドライブとしてPC-9801を動かす準備ができました。

まず，PC-9801の電源を入れてください。次に，今回試作した，プログラム「R.EXE」を実行します。パラメータには仮想ドライブとして使用したいドライブ番号を指定します。

<例>

```
A>R -DC [Cr]
```

ここで気をつけてほしいのは共有を開始したいドライブ番号を指定するということです。たとえば，A:，B:，C:という3つのドライブがあったとします。A:はハードディスク，B:とC:はフロッピーディスクです。このときに，B:ドライブから仮想化したいなら，

```
A>R -DB [Cr]
```

として起動します。これより，B:および続くC:ドライブが仮想ディスクとして主機に登録，使用できるようになります。

なぜ，このような仕様にしたかというと，通常はハードディスクのドライブ番号はフロッピーより前にくるためです。ハードディスクを仮想化するのは危険なため，

移植のテクニック2

今回新たに作ったプログラムは主に，XCに存在していて，Quick-C(MS-C)に存在しない関数ばかりです。これまでデータ型やデータ並びとは別に，XCにのみ存在している関数をPC-9801上で用意してあげる必要があります。次にリストアップするのがそういった関数です。XC互換の関数を作ることにより，移植を容易にすることができます。このように関数にまとめた場合は機種に依存したコードを特定の関数に集約することができるのでスマートに移植することができます。

int LOF232C()　RS-232C受信文字数を調べる
int INP232C()　RS-232Cより1文字受信
void OUT232C() RS-232Cへ1文字送信
int GETDPB()　DPBブロックを取得する
void DISKRED() ディスク読み取り
void DISKWRT() ディスク書き込み
long DRVCTRL() ドライブ情報取得

これらの機能は，ほとんどすべてPC-9801上でなんらかの形で用意されています。ですから，各関数の中にこの機能を記述してあげればよいわけです。

腕に自信のある人は，リスト2中612行目以降を参照してください。ここには上記の関数群が記述されています。

また，次の関数は今回特別に作成したものです。

int_rs_inz()　RS-232Cを9600bpsに初期化
void xchg2()　2バイトデータの上下を並べ替える
void xchg4()　4バイトデータの上下を並べ替える

_rs_inz()は，PC-9801のBIOS機能を使ってRS-232Cを初期化しています。これにより，RS-232Cを使えるようにしています。前回までのX68000版では，初期化はHuman68kの「SPEED.X」を使っていたので，このような処理がありませんでした。

xchg2()，xchg4()は本文中で記述している，i8086系とMC68000系のデータ並びの相違を吸収するものです。2バイトないし4バイトのデータの上下を入れ換えます。

これはRS-232C経由でX68000より受け取ったデータをこの関数でPC-9801用に変換するときに使います。また，逆にPC-9801用のデータをX68000に送るときに，あらかじめX68000用に変換してあげるために使います。

(クラッシュした場合は) 明示的に仮想化を開始したドライブを指定する形にしました。

ですから，すべてを仮想ドライブとして登録したい場合は，

A>R -DA [Cr]

としてください。

次に，主機の電源を入れます。これにより，仮想ドライブは主機の後ろ側に追加登録を行います。

ですから，主機の最終ドライバ番号が「D:」であれば，「E:」から割り振られます。

このプログラムを終了したいときには，エスケープキーを押してください。強制終了します。

また，このときに，もしも従機の電源が入っていない，または「R.EXE」が起動されていなければ，この仮想ドライバシステムは登録されません。

2DDディスクが読めるか?

普通のX68000ではサポートされていませんが，PC-9801では2DDのフロッピーディスクが利用できます。今回の仮想ドライブにPC-9801の2DDのフロッピーディスクを入れた場合どうなるでしょうか。

理屈のうえでは，ディスクの管理はBPBパラメータテーブルに従いHuman68kが制御するので，仮想ドライブのBPBテーブルをきちんと設定して，これに従って動かす分には支障ないはずです。

結論から先にいいますと，2DDの720Kバイトフォーマットのフロッピーをセットしたところアクセスできました。

テストとして，Human68kの「COMMAND.X」をPC-9801の2DDに転送。そして，再びこれを，X68000のFDに転送し，本来の「COMMAND.X」とファイル比較したところ，同一の結果が出ました。これより，PC-9801の2DDドライブをそのまま読めることが確認されました。

ただ，残念なことに，理由は不明ですが，PC-9801からX68000のRAMディスクにはなぜか転送できませんでした。まだまだクリアしなくてはいけない技術ポイントがあるのかもしれません。

ハードディスクに関しては，今回は怖くて詳しくテストしていません。PC-9801のハードディスクのディレクトリを取る程度の簡単な実験しかしていません。

読者の方が，このシステムを動かすときは慎重に行ってください。

通常のアプリケーションプログラムですと，プログラムのバグ，予期せぬ事態によりディスクが壊れる場合，たいていは書き込み参照中のファイルがダメになる程度です。しかし，当システムのようなデバイスドライバはディスクのファイル管理メカニズムそのものですので，障害が起きたときには全ファイルが破壊されることにもなりかねません。仮に，ファイル情報が正しく書き込まれたとしても，ディスク中のどこに存在するかを管理するFAT (ファイルアロケーションテーブル) が壊れれば，もはや，すべては失われたも同様です。データがディスク上に存在しても，位置がわからなければ利用できないからです。

次回の予定

ここのところ，実験開発を少し進めすぎたと思います。仮想ドライブシステムを，どうにか構築することができましたが，評価テストが不足しているように感じています。

ですから，次回は主に評価テストを行いたいと思います。X68000←→X68000およびX68000←→PC-9801で接続して，利用できるメディア，できないメディア，また，仮想ドライブの作動するプログラムなどのレポートを行いたいと思います。

また，PC-9801上のフロッピーを途中から2DDと2HDを入れ換えた場合などいろいろな局面が考えられます。これらを1つひとつ検証していきたいと思います。

製品化について(仮称Z-Link)

現在，当連載で実験した結果を製品になんらかの形でフィードバックしたいと考えています。

基本仕様としては，X68000を主機とした仮想ドライブシステムです。従機にはX68000，PC-9801，DOS/V機をぶらさげられるようにします。

転送速度は，次のようなものを考えています。

X68000↔X68000	76,800bps
X68000↔DOS/V	38,400bps
X68000↔PC9801(10MHz系)	76,800bps
X68000↔PC9801(8MHz系)	9,600bps

販売形態は直接販売を取るか，流通を通すかで価格もずいぶんと異なってきます。できれば，ケーブル一式すべてをつけて9,800円〜19,800円以内に収まるようにしてパッケージ化できれば，と思っています。技術的な問題も山積みしていますが，8月頃にパッケージ化できるのではないかと観測しています。

リスト1

```
=============== d1.c ==================
 1: #define  DEBUG  1
 2:
 3:
 4: #include  <doslib.h>
 5: #include  <stdio.h>
 6: #include  <time.h>
 7:
 8: extern  short  d_tim;
 9: extern  short  d_dte;
10: extern  char   d_dat[];
11:
12: /*
13: extern  struct  BPBPOI  *inittbl;
14: */
15:
16:
17: extern  char  inittbl;
18: extern  char  inittblend;
19: extern  unsigned short  rdmaxr;
20:
21:
22: #define  SRAM    0xed0000
23: #define  SRAMMD  0xed002d
24: #define  SYS00d  0xe8e00d
25:
26: #define  TIMEOUT  5L
27:
28:
29:
30: #define  REQLEN   0
31: #define  UNITCD   1
32: #define  COMCOD   2
33: #define  ERRLOW   3
34: #define  ERRHIGH  4
35:
36: #define  MXUNIT  13
37: #define  DEVEND  14
38: #define  BPBPOI  18
39: #define  BDEVNO  22
40:
41: #define  DISKID  13
```

▶学校のUNIXワークステーションが速くなった。そろそろX68000も速いヤツじゃないとガマンできなくなってきたぞ。
筒井　圭一朗(20)埼玉県

```
 42: #define  DISKFG  14
 43:
 44: #define  DMAADR  14
 45: #define  DMALEN  18
 46: #define  STAREC  24
 47: #define  GETDAT  13
 48:
 49: char  *_devmes = "C: $ED0400-15K";
 50: char  *_mesini = ")初期化しました ¥n¥r";
 51: char  *_mesnoi = ")初期化しませんでした ¥n¥r";
 52:
 53: int  dskinp();
 54: int  dskout();
 55: int  dskini();
 56: int  mediac();
 57: int   dskpaw0();
 58: int   dskpaw1();
 59: int   dskpaw2();
 60: int   dskotv();
 61: long    _abs();
 62: void  _rs_buf_clr();
 63: int notcom();
 64:
 65: int qout232c( int c );
 66:
 67: /*----- d3.c func -----*/
 68: int  blk_in();
 69: int  blk_inl();
 70: int  blk_out();
 71: int  blk_outl();
 72: void    _rs_buf_clr();
 73:
 74:
 75: struct REQ_HED {
 76:    char  reqlen;
 77:    char  unitcd;
 78:    char  comcod;
 79:    char  errlow;
 80:    char  errhigh;
 81: };
 82:
 83: struct REQ_INI {
 84:   char  reqlen;     /* リクエストヘッダの長さ */
 85:   char  unitcd;     /* ユニットコード */
 86:   char  comcod;     /* コマンドコード */
 87:   char  errlow;     /* エラーコードその1 */
 88:   char  errhigh;    /* エラーコードその2 */
 89:   char  rsv[8];     /* 予約領域 */
 90:         /* -------------------------    */
 91:   char  mxunit;     /* ユニット数 */
 92:   char  *devend;    /* デバイスドライバ制御プログラムの終了アドレス*/
 93:   char  *bpbpoi;    /*struct  BPBPOI *bpbpoi; BPBテーブルアドレス*/
 94:   char  bdevno;     /* ブロックデバイス番号(0=A*) */
 95: };
 96:
 97:
 98: struct REQ_CHG {
 99:   char  reqlen;     /* リクエストヘッダの長さ */
100:   char  unitcd;     /* ユニットコード      */
101:   char  comcod;     /* コマンドコード      */
102:   char  errlow;     /* エラーコードその1 */
103:   char  errhigh;    /* エラーコードその2 */
104:   char  rsv[8];     /* 予約領域        */
105:         /* -------------------------    */
106:   char  diskid;     /* よくわかんない<現在未使用> */
107:   long  diskfg;     /* よくわかんない 資料ではバイト型*/
108: };
109:
110:
111: struct REQ_RW {
112:   char  reqlen;     /* リクエストヘッダの長さ*/
113:   char  unitcd;     /* ユニットコード      */
114:   char  comcod;     /* コマンドコード      */
115:   char  errlow;     /* エラーコードその1 */
116:   char  errhigh;    /* エラーコードその2  */
117:   char  rsv[8];     /* 予約領域        */
118:         /* -------------------------    */
119:   char  diskid;     /* よくわかんない<現在未使用>*/
120:   char  *dmaadr;    /* 転送バッファ       */
121:   long  dmalen;     /* 転送セクター数     */
122:   char  starec00[2]; /* アクセスセクタ番号上位2byte*/
123:   long  starec;     /* アクセスセクタ番号     */
124: };
125:
126:
127: struct REQ_CTRL {
128:   char  reqlen;     /* リクエストヘッダの長さ  */
129:   char  unitcd;     /* ユニットコード      */
130:   char  comcod;     /* コマンドコード      */
131:   char  errlow;     /* エラーコードその1     */
132:   char  errhigh;    /* エラーコードその2     */
133:   char  rsv[8];     /* 予約領域        */
134:         /* -------------------------    */
135:   char  getdat;
136: };
137:
138:
139: struct REQ_RW req_hed;
140:
141: #if DEBUG
142:
143: char  buf[2048];
144: /****************************************************************
145:    main() テスト用のメインルーチン
146:    エントリー(entry)は全プログラムの入り口と解釈すること
147:    ASMを強引にハンドディスコンパイル
148: ****************************************************************/
149: void main()
150: {
151:   int  sts;
152:   int  s;
153:
154:   for( s=0 ; s<10 ; s++ ) {
155:     req_hed.reqlen = sizeof(struct REQ_CHG); /*リクエストヘッダの長さ*/
156:     req_hed.comcod = s;
157:
158:     switch( s ) {
159:       case 0:
160:         sts = dskini( &req_hed );
161:         break;
162:
163:       case 1:
164:         sts = mediac( &req_hed );
165:         break;
166:
167:       case 4:
168:         req_hed.reqlen = sizeof(struct REQ_RW);
169:         req_hed.dmaadr = buf;
170:         req_hed.starec00[0] = req_hed.starec00[1] = 0;
171:         req_hed.dmalen = 1L;
172:         req_hed.starec = 1L;
173:
174:         sts = dskinp( &req_hed );
175:         break;
176:
177:       case 5:
178:         req_hed.reqlen = sizeof(struct REQ_CTRL);
179:         sts = dskctrl( &req_hed );
180:         break;
181:
182:       case 8:
183:         req_hed.reqlen = sizeof(struct REQ_RW);
184:         buf[0] = 9;
185:         req_hed.dmaadr = buf;
186:         req_hed.starec00[0] = req_hed.starec00[1] = 0;
187:         req_hed.dmalen = 17L;
188:         req_hed.starec = 1L;
189:         sts = dskout( &req_hed );
190:         break;
191:
192:       case 9:
193:         req_hed.reqlen = sizeof(struct REQ_RW);
194:         sts = dskotv( &req_hed );
195:         break;
196:
197:       default:
198:         sts = notcom(&req_hed);         /* 未使用命令*/
199:         break;
200:     }
201:   }
202: }
203: #endif
204:
205:
206:
207: /****************************************************************
208:    diskent  世に言うエントリールーチン (舶来語)
209:    エントリー(entry)は全プログラムの入り口と解釈すること
210:    ASMを強引にハンドディスコンパイル
211: ****************************************************************/
212: void dskent( req_hed )
213: struct REQ_HED *req_hed;
214: {
215:   int  sts;
216:
217:   switch( req_hed->comcod ) {
218:     case 0:
219:       sts = dskini( req_hed );
220:       break;
221:     case 1:
222:       sts = mediac( req_hed );
223:       break;
224:     case 2:
225:       sts = notcom(req_hed);        /* 未使用命令*/
226:       break;
227:     case 3:
228:       sts = notcom(req_hed);        /* 未使用命令*/
229:       break;
230:     case 4:
231:       sts = dskinp( req_hed );
232:       break;
233:     case 5:
234:       sts = dskctrl( req_hed );
235:       break;
236:     case 6:
237:       sts = notcom(req_hed);        /* 未使用命令*/
238:       break;
239:     case 7:
240:       sts = notcom(req_hed);        /* 未使用命令*/
241:       break;
242:     case 8:
243:       sts = dskout( req_hed );
244:       break;
245:     case 9:
246:       sts = dskotv( req_hed );
247:       break;
248:     case 10:
249:       sts = notcom(req_hed);        /* 未使用命令    */
250:       break;
251:     case 11:
252:       sts = notcom(req_hed);        /* 未使用命令    */
253:       break;
254:     case 12:
255:       sts = notcom(req_hed);        /* 未使用命令    */
256:       break;
257:     default:
258:       sts = notcom(req_hed);        /* 未使用命令    */
259:       break;
260:   }
261: }
262:
263:
264: /****************************************************************
265:    notcom  未使用命令
266:    ASMを強引にハンドディスコンパイル
267: ****************************************************************/
268: int notcom(req)
269: struct REQ_HED *req;
270: {
271:   req->errlow = 0x03;         /* 下位バイトエラーコード格納*/
272:   req->errhigh = 0x50;        /* 上位バイトエラーコード格納*/
273:   return( 0 );
274: }
275:
276:
277:
```

▶なんと、大学で処分寸前だったX68000 ACEを手に入れました。しかも2台。もう大学やめてもいいです(冗談2%)。里子に出したあとだけに……。　和田　智(18)長野県

```
278:
279: /*****************************************************************
280:   mediac  メディア交換処理(RAMなので未使用のはず)
281: *****************************************************************/
282: int mediac( req_chg )
283: struct REQ_CHG *req_chg;
284: {
285:   int  sts;
286:   int  n;
287:   char  c;
288:   req_chg->diskfg = 1L;           /* エラーコードセット  */
289:   n = (int)(req_chg->reqlen);
290:
291:
292:   if( qout232c( 'S' ) ) {         /* 送信開始コード送出  */
293:     ;
294:   }
295:   else if( (sts=blk_in( &c, 1) )) {        /* 応答コード受信*/
296:     ;
297:   }
298:   else if( c != 'A' ) {
299:     ;
300:   }
301:   else if( (sts=blk_out( &n, sizeof(n) ))) {/*リクエストヘッダ長送信 */
302:     ;
303:   }
304:   else if( (sts=blk_out( req_chg, req_chg->reqlen ))) { /*リクエストヘッダ送信 */
305:     ;
306:   }
307:   else if( (sts=blk_in( &(req_chg->diskfg), sizeof(req_chg->diskfg) ))) {
308:     req_chg->diskfg = 1L;         /* エラーコードセット  */
309:   }
310:   return( 0 );
311: }
312:
313:
314:
315:
316: /*****************************************************************
317:   dskinp  ディスク入力処理(?)
318:   ASMを強引にハンドディスコンパイル
319: *****************************************************************/
320: int dskinp( req )
321: struct REQ_RW *req;
322: {
323:   int   sts;
324:   int   n;
325:   int   len;
326:   int   wk;
327:   char  *p;
328:   char  c;
329:
330:   req->errlow = 0x03;           /* 下位バイトエラーコード格納*/
331:   req->errhigh = 0x50;          /* 上位バイトエラーコード格納*/
332:
333:   n = (int)(req->reqlen);
334:
335:   if( qout232c( 'S' ) ) {               /* 送信開始コード送出  */
336:     ;
337:   }
338:   else if( (sts=blk_in( &c, 1) )) {        /* 応答コード受信  */
339:     ;
340:   }
341:   else if( c != 'A' ) {
342:     ;
343:   }
344:   else if( (sts=blk_out( &n, sizeof(n) ))) {/*リクエストヘッダ長送信*/
345:     ;
346:   }
347:   else if( (sts=blk_out( req, req->reqlen ))) {/*リクエストヘッダ送信*/
348:     ;
349:   }
350:   else if( (sts=blk_in( &len, 4 ))) {
351:     ;
352:   }
353:   else {
354:     p = req->dmaadr;
355:
356:     while( len>0 ) {
357:       wk = len;
358:       if( wk>16384L ) {
359:         wk = 16384L;
360:       }
361:
362:       if( (sts=blk_in( p, wk ))) {
363:         break;
364:       }
365:
366:       p += 16384;
367:       len -= 16384;
368:     }
369:
370:     if( len<=0 ) {
371:       req->errlow = 0;          /* 下位バイトエラーコード格納  */
372:       req->errhigh = 0;         /* 上位バイトエラーコード格納  */
373:       sts = 0;
374:     }
375:   }
376:
377:   return( sts );
378: }
379:
380:
381:
382:
383: /*****************************************************************
384:   dskotv  出力&ベリファイ処理(?)
385:   ASMを強引にハンドディスコンパイル
386: *****************************************************************/
387: int dskotv( req )
388: struct REQ_RW *req;
389: {
390:   return( dskout( req ) );
391: }
392:
393:
394:
395: /*****************************************************************
396:   dskout  ディスク出力処理(?)
397:   ASMを強引にハンドディスコンパイル
398: *****************************************************************/
399: int dskout( req )
400: struct REQ_RW *req;
401: {
402:   int   sts;
403:   int   n;
404:   int   len;
405:   int   wk;
406:   char  *p;
407:   char  c;
408:   short int  *b_no;
409:   char  **wk1;
410:   char  *bpb;
411:
412:   req->errlow = 0x03;           /* 下位バイトエラーコード格納  */
413:   req->errhigh = 0x50;          /* 上位バイトエラーコード格納  */
414:
415:   n = (int)(req->reqlen);
416:
417:   if( qout232c( 'S' ) ) {       /* 送信開始コード送出  */
418:     ;
419:   }
420:   else if( (sts=blk_in( &c, 1) )) {   /* 応答コード受信 */
421:     ;
422:   }
423:   else if( c != 'A' ) {
424:     ;
425:   }
426:   else if( (sts=blk_out( &n, sizeof(n) ))) {/*リクエストヘッダ長送信*/
427:     ;
428:   }
429:   else if( (sts=blk_out( req, req->reqlen ))) {/*リクエストヘッダ送信*/
430:     ;
431:   }
432:   else {
433:
434:     wk1 = (char**)(&inittbl);
435:     bpb = *wk1;
436:     bpb += (req->unitcd * 12);
437:     b_no = (short int*)bpb;
438:
439:     len = req->dmalen;  /* len = 転送バイト数算出 */
440:     len *= (*b_no);     /* len = dmalen * 1024   */
441:
442:     if( (sts=blk_out( &len, 4 ))) {  /* データ長送信*/
443:       ;
444:     }
445:     else {
446:       p = req->dmaadr;
447:
448:       while( len>0 ) {
449:         wk = len;
450:         if( wk>16384L ) {
451:           wk = 16384L;
452:         }
453:
454:         if( (sts=blk_out( p, wk ))) {  /* データ長送信*/
455:           break;
456:         }
457:
458:         c=INP232C();
459:
460:         p += 16384;
461:         len -= 16384;
462:       }
463:
464:       if( len<=0 ) {
465:         if( (sts=blk_in( &req->errlow, 2 ))) {/*エラーコードセット*/
466:           ;
467:         }
468:         else if( req->errlow || req->errhigh ) {
469:           ;
470:         }
471:         else {
472:           req->errlow = 0;          /* 下位バイトエラーコード格納*/
473:           req->errhigh = 0;         /* 上位バイトエラーコード格納*/
474:           sts = 0;
475:         }
476:       }
477:     }
478:   }
479:
480:   return( sts );
481: }
482:
483:
484:
485:
486: /*****************************************************************
487:   dskctrl  ディスクコントロール
488: *****************************************************************/
489: int dskctrl( req )
490: struct REQ_CTRL *req;
491: {
492:   int  sts;
493:   int  n;
494:   int  len;
495:   char  c;
496:
497:   req->errlow = 0x03;           /* 下位バイトエラーコード格納 */
498:   req->errhigh = 0x50;          /* 上位バイトエラーコード格納 */
499:
500:   n = (int)(req->reqlen);
501:
502:   if( qout232c( 'S' ) ) {               /* 送信開始コード送出*/
503:     ;
504:   }
505:   else if( (sts=blk_in( &c, 1) )) {  /* 応答コード受信 */
506:     ;
507:   }
508:   else if( c != 'A' ) {
509:     ;
510:   }
511:   else if( (sts=blk_out( &n, sizeof(n) ))) {   /*リクエストヘッダ長送信*/
512:     ;
513:   }
```

▶6月号には(善)氏がどこにもいない。(善)氏のいないOh!Xは気のぬけたコーラのようなものだ。しかし6月号はレベルが高い。　佐怒賀　英一(26)神奈川県

```
514:   else if( (sts=blk_out( req, req->reqlen ))) {/*リクエストヘッダ送信*/
515:     ;
516:   }
517:   else if( (sts=blk_in( req, req->reqlen ))) { /*リクエストヘッダ受信*/
518:     ;
519:   }
520:   else {
521:     sts = 0;
522:   }
523:
524:   return( sts );
525: }
526:
527:
528: /*****************************************************************
529:   dskini  ディスク初期化ルーチン
530: *****************************************************************/
531: int dskini( req )
532: struct REQ_INI *req;
533: {
534:   int   sts;
535:   int   n;
536:   char  u_no;
537:   char  drv_flg;
538:   char  **wk;
539:   char  *bpb;
540:   char  c;
541:
542:   _rs_buf_clr();               /* rs232c buf clr  */
543:
544:   n = (int)(req->reqlen);
545:   req->errlow = 0x03;            /* 下位バイトエラーコード格納*/
546:   req->errhigh = 0x50;           /* 上位バイトエラーコード格納*/
547:
548:   u_no = 0;
549:   wk = (char**)(&inittbl);
550:   bpb = *wk;
551:
552:   if( qout232c( 'S' ) ) {               /* 送信開始コード送出  */
553:     ;
554:   }
555:   else if( (sts=blk_in( &c, 1) )) {  /* 応答コード受信 */
556:     ;
557:   }
558:   else if( c != 'A' ) {
559:     ;
560:   }
561:   else if( (sts=blk_out1( &n, sizeof(n) ))) { /*リクエストヘッダ長送信*/
562:     ;
563:   }
564:   else if( (sts=blk_out( req, req->reqlen ))) {/* リクエストヘッダ送信*/
565:     ;
566:   }
567:   else {
568:     PRINT( (unsigned char*)"¥n¥r仮想ドライブシステム登録開始¥n¥r" );
569:
570:     while( 1 ) {
571:       if( (sts=blk_in( &drv_flg, 1 ))) { /* bps status flg受信*/
572:         break;
573:       }
574:       else if( drv_flg == -1 ) {          /* normal end*/
575:         if( u_no==0 ) {
576:           break;
577:         }
578:
579:         req->errlow = 0;          /* 下位バイトエラーコード格納*/
580:         req->errhigh = 0;         /* 上位バイトエラーコード格納*/
581:
582:         req->mxunit = u_no;      /* 最大unit  セット*/
583:         req->bpbpoi = &inittbl;     /*BPBテーブルアドレスセット*/
584:         req->devend = &inittblend; /*デバイステーブルエンドアドレスセット*/
585:         break;
586:       }
587:       else {
588:         if( (sts=blk_in( bpb, 12 ))) { /*bpb table受信*/
589:           break;
590:         }
591:
592:         bpb += (char*)(12);
593:         u_no ++;
594:         PRINT( (unsigned char*)("  仮想ドライブを1基追加 ¥n¥r") );
595:       }
596:     }
597:   }
598:
599:   return( 0 );
600: }
601:
602: int qout232c( int c )
603: {
604:   int  j;
605:   int  i;
606:   int  sts;
607:
608:   sts = -1;
609:   OUT232C( c );
610:
611:   for( j=0 ; j<3 ; j++ ) {
612:     for( i=0 ; i<7000 ; i++ ) {
613:       if( LOF232C() ) {
614:         sts = 0;
615:         break;
616:       }
617:     }
618:
619:     if( sts==0 ) {
620:       break;
621:     }
622:
623:     OUT232C( c );
624:
625:   }
626:
627:   return( sts );
628: }
```

リスト2

```
===============  R.C  ==================
  1: #define          X68     0
  2: #define          PC98    1
  3: #define          IBM     2
  4: #define          SYSTEM  1
  5:
  6: #define  DEBUG  1
  7: #define  ESC  0x1b
  8:
  9: #if SYSTEM==X68
 10: #include  <doslib.h>
 11: #include  <stdio.h>
 12: #include  <time.h>
 13: #define          blk_in_x(a,b)   blk_in(a,b)
 14: #define          blk_out_x(a,b)   blk_out(a,b)
 15: #define  XCHG2(a)
 16: #define  XCHG4(a)
 17: #endif
 18:
 19: #if SYSTEM==PC98 || SYSTEM==IBM
 20: #pragma pack(1)
 21: #include         <stdlib.h>
 22: #include         <dos.h>
 23: #include         <sys¥types.h>
 24: #include         <sys¥stat.h>
 25:
 26:
 27: typedef  long      INT;
 28: typedef  unsigned long  UINT;
 29: typedef  short     WORD;
 30: typedef  unsigned short  UWORD;
 31: typedef  char      BYTE;
 32: typedef  unsigned char  UBYTE;
 33: typedef  void      VOID;
 34: typedef  int       DEFAULT;
 35: typedef  int       BOOLEAN;
 36: typedef  long      LONG;
 37: typedef  unsigned long  ULONG;
 38:
 39:
 40:
 41: struct  DPBPTR  {
 42:   UBYTE  drive;
 43:   UBYTE  unit;
 44:   UWORD  byte;
 45:   UBYTE  sec;
 46:   UBYTE  shift;
 47:   UWORD  fatsec;
 48:   UBYTE  fatcount;
 49: #if SYSTEM==X68
 50:   UBYTE  fatlen;
 51: #endif
 52:   UWORD  dircount;
 53:   UWORD  datasec;
 54:   UWORD  maxfat;
 55: #if SYSTEM==PC98 || SYSTEM==IBM
 56:   UBYTE  fatlen;
 57: #endif
 58:   UWORD  dirsec;
 59:   INT  driver;
 60:   UBYTE  id;
 61:   UBYTE  flg;
 62:
 63:   char  next[4];  /*struct  DPBPTR  *next;*/
 64:   UWORD  dirfat;
 65:   UBYTE  dirbuf[64];
 66: };
 67:
 68: int      _rs_inz();
 69: int  LOF232C();
 70: int  INP232C();
 71: void  OUT232C();
 72: int      GETDPB( int drv, struct DPBPTR  *d );
 73: void DISKRED( unsigned char* _rw_buf, int drv, long rec, long len );
 74: void DISKWRT( unsigned char* _rw_buf, int drv, long rec, long len );
 75: long DRVCTRL( long mode, int drv );
 76: int  blk_in_x();
 77: int  blk_out_x();
 78: void xchg2( char *data );
 79: void xchg4( char *data );
 80:
 81: #define XCHG2(a)         xchg2((char*)(a))
 82: #define XCHG4(a)         xchg4((char*)(a))
 83:
 84: #endif
 85:
 86:
 87: #define  TIMEOUT  5L
 88:
 89:
 90: #define  REQLEN  0
 91: #define  UNITCD  1
 92: #define  COMCOD  2
 93: #define  ERRLOW  3
 94: #define  ERRHIGH  4
 95:
 96: #define  MXUNIT  13
 97: #define  DEVEND  14
 98: #define  BPBPOI  18
 99: #define  BDEVNO  22
100:
101: #define  DISKID  13
102: #define  DISKFG  14
103:
104: #define  DMAADR  14
105: #define  DMALEN  18
```

▶ちゃだワ恒例の酒井さんの自画像が，今年はなかったので寂しかったのです。が，ひと月遅れて見ることができ，ちゃだワがきたなあという感じです(春がきたことを知らせる風物詩?)。
黒武者　健一(24)神奈川県

```
106: #define  STAREC  24
107: #define  GETDAT  13
108:
109: void   rcv_ent();
110: int  r_dskinp();
111: int  r_dskout();
112: int  r_dskini();
113: int  r_mediac();
114: int   r_dskpaw0();
115: int   r_dskpaw1();
116: int   r_dskpaw2();
117: int   r_dskotv();
118:
119: long   _abs();
120: void _rs_buf_clr();
121: int notcom();
122:
123: /*----- d3.c func -----*/
124: int  blk_in();
125: int  blk_in1();
126: int  blk_out();
127: int  blk_out1();
128: void   _rs_buf_clr();
129:
130:
131: struct REQ_HED {
132:   char  reqlen;
133:   char  unitcd;
134:   char  comcod;
135:   char  errlow;
136:   char    errhigh;
137: };
138:
139: struct REQ_INI {
140:   char  reqlen;   /* リクエストヘッダの長さ  */
141:   char  unitcd;   /* ユニットコード     */
142:   char  comcod;   /* コマンドコード     */
143:   char  errlow;   /* エラーコードその1     */
144:   char  errhigh;  /* エラーコードその2     */
145:   char  rsv[8];   /* 予約領域       */
146:           /* -------------------------   */
147:   char  mxunit;   /* ユニット数       */
148:   char  devend[4];/*(*devend;) デバイスドライバ制御プログラム終了アドレス*/
149:   char  bpbpoi[4];/* (struct  BPBPOI *bpbpoi; BPBテーブルアドレス*/
150:   char  bdevno;   /* ブロックデバイス番号(0=A*)  */
151: };
152:
153:
154: struct REQ_CHG {
155:   char  reqlen;    /* リクエストヘッダの長さ  */
156:   char  unitcd;    /* ユニットコード     */
157:   char  comcod;    /* コマンドコード     */
158:   char  errlow;    /* エラーコードその1     */
159:   char  errhigh;   /* エラーコードその2     */
160:   char  rsv[8];    /* 予約領域       */
161:           /* -------------------------   */
162:   char  diskid;    /*よくわかんない<現在未使用> */
163:   long  diskfg;    /*よくわかんない 資料ではバイト型*/
164: };
165:
166:
167: struct REQ_RW {
168:   char  reqlen;    /* リクエストヘッダの長さ  */
169:   char  unitcd;    /* ユニットコード     */
170:   char  comcod;    /* コマンドコード     */
171:   char  errlow;    /* エラーコードその1     */
172:   char  errhigh;   /* エラーコードその2     */
173:   char  rsv[8];    /* 予約領域       */
174:           /* -------------------------   */
175:   char  diskid;    /* よくわかんない<現在未使用>*/
176:   char  dmaadr[4]; /* (*dmaadr;) 転送バッファ  */
177:   long  dmalen;    /* 転送セクター数     */
178:   char  starec00[2]; /* アクセスセクタ番号上位2byte*/
179:   long  starec;    /* アクセスセクタ番号     */
180: };
181:
182:
183: struct REQ_CTRL {
184:   char  reqlen;    /* リクエストヘッダの長さ*/
185:   char  unitcd;    /* ユニットコード     */
186:   char  comcod;    /* コマンドコード     */
187:   char  errlow;    /* エラーコードその1*/
188:   char  errhigh;   /* エラーコードその2*/
189:   char  rsv[8];    /* 予約領域       */
190:           /* ------------------------- */
191:   char  getdat;
192: };
193:
194:
195: struct REQ_CHG req_hed;
196:
197: int  _drv;
198: int     _byte[10];
199: /*****************************************************************
200:   main() テスト用のメインルーチン
201:   エントリー(entry)は全プログラムの入り口と解釈すること
202: *****************************************************************/
203: void main( argc, argv )
204: int  argc;
205: unsigned char *argv[];
206: {
207:   int  sts;
208:   int  c;
209:   int  i;
210:
211:   _rs_inz();
212:   _drv = 4;                              /* ドライブNoセット = D: */
213:   printf( "仮想DISK system V1.1 ¥n" );
214:   printf( "[ESC] exit ¥n" );
215:
216:
217:
218:   for( i=1 ; i<argc ; i++ ) {
219:     if( memcmp( argv[i], "-d", 2 )==0 || memcmp( argv[i], "-D", 2 )==0 ) {
220:       _drv = (int)(argv[i][2] - 'A');
221:       if( argv[i][2] >= 'a' ) {
222:         _drv = (int)(argv[i][2] - 'a');
223:       }
224:
225:       _drv ++;
226:
227:     }
228:     else {
229:       printf( "R [-H] ¥n" );
230:       printf( "-H   ······ help message ¥n" );
231:       printf( "-D   ······ 仮想化開始Kのドライブを指定 ¥n" );
232:       printf( "              例: R -DC ¥n" );
233:       printf( "                 Cドライブから仮想化する場合 ¥n" );
234:
235:       exit(1);
236:     }
237:   }
238:
239:   _rs_buf_clr();          /* 通信バッファクリア    */
240:
241:   while( 1 ) {
242:
243:     if( LOF232C() ) {         /* 同調機構、受信側でとりこぼし */
244:       c = INP232C();
245:
246:       if( c!='S' ) {
247:         _rs_buf_clr();  /* 通信バッファクリア    */
248:       }
249:       else if( (sts=blk_out( "A", 1 )) ) {
250:         _rs_buf_clr();  /* 通信バッファクリア    */
251:       }
252:       else {
253:         rcv_ent();
254:       }
255:     }
256:
257:     while( kbhit() ) {
258:       if( getch()==ESC ) {
259:         exit(0);
260:       }
261:     }
262:
263:   }
264: }
265:
266:
267: char  _rq_buf[1024];
268: char  _rw_buf[16384];
269: struct REQ_HED *_rq = (struct REQ_HED*)_rq_buf;
270:
271: /*****************************************************************
272:   rcv_ent  通信パケット処理
273: *****************************************************************/
274: void rcv_ent()
275: {
276:   int  sts;
277:   INT  n;
278:
279:   if( (sts=blk_in_x( &n, sizeof(n)  ))) {  /* パケット長取得*/
280:     ;
281:   }
282:   else if( (sts=blk_in( _rq, n  ))) {  /* パケット取得  */
283:     ;
284:   }
285:
286:   switch( _rq->comcod ) {
287:     case 0:
288:       sts = r_dskini( _rq );
289:       break;
290:
291:     case 1:
292:       sts = r_mediac( _rq );
293:       break;
294:
295:     case 2:
296:       sts = r_notcom( _rq );      /* 未使用命令    */
297:       break;
298:
299:     case 3:
300:       sts = r_notcom( _rq );      /* 未使用命令    */
301:       break;
302:
303:     case 4:
304:       sts = r_dskinp( _rq );
305:       break;
306:
307:     case 5:
308:       sts = r_dskctrl( _rq );
309:       break;
310:
311:     case 6:
312:       sts = r_notcom( _rq );      /* 未使用命令    */
313:       break;
314:
315:     case 7:
316:       sts = r_notcom( _rq );      /* 未使用命令    */
317:       break;
318:
319:     case 8:
320:       sts = r_dskout( _rq );
321:       break;
322:
323:     case 9:
324:       sts = r_dskotv( _rq );
325:       break;
326:
327:     case 10:
328:       sts = r_notcom( _rq );      /* 未使用命令    */
329:       break;
330:
331:     case 11:
332:       sts = r_notcom( _rq );      /* 未使用命令    */
333:       break;
334:
335:     case 12:
336:       sts = r_notcom( _rq );      /* 未使用命令    */
337:       break;
338:
339:     default:
340:       sts = r_notcom( _rq );      /* 未使用命令    */
341:       break;
```

▶ゲームもいらなきゃテレカもいらぬ，あたしゃやっぱり湯飲みがほしいってね。いまだに予約しなかったことを後悔している。　*井村　英二(23)滋賀県*

```
342:   }
343: }
344:
345:
346: /*****************************************************************
347:   notcom  未使用命令
348: *****************************************************************/
349: int r_notcom(req)
350: struct REQ_HED *req;
351: {
352:   req->errlow = 0x03;      /* 下位バイトエラーコード格納*/
353:   req->errhigh = 0x50;     /* 上位バイトエラーコード格納*/
354:   return( 0 );
355: }
356:
357:
358:
359:
360: /*****************************************************************
361:   r_mediac  メディア交換処理(RAMなので未使用のはず)
362: *****************************************************************/
363: int r_mediac( req_chg )
364: struct REQ_CHG *req_chg;
365: {
366:   int  sts;
367:   int  n;
368:
369:   req_chg->diskfg = 0L;          /* stsコードセット  */
370:
371:
372:   if( (sts=blk_out( &(req_chg->diskfg), sizeof(req_chg->diskfg) ))) { /*リクエ
ストヘッダ送信 */
373:     ;
374:   }
375:
376:   return( 0 );
377: }
378:
379:
380:
381:
382: /*****************************************************************
383:   r_dskinp  ディスク入力処理(?)
384: *****************************************************************/
385: int r_dskinp( req )
386: struct REQ_RW *req;
387: {
388:   int   sts;
389:   int   n;
390:   long  len;
391:   long  rec;
392:   long  wk;
393:   long  wk1;
394:   char  *p;
395:
396:   req->errlow = 0;             /* 下位バイトエラーコード格納 */
397:   req->errhigh = 0;            /* 上位バイトエラーコード格納 */
398:
399:   XCHG4( &(req->dmalen) );          /* もしDOSなら配列変換 */
400:   XCHG4( &(req->starec) );          /* もしDOSなら配列変換 */
401:
402:   len = req->dmalen * (long)_byte[req->unitcd];
403:   rec =req->starec;
404:
405:   rec >>= 16;
406:
407:
408:   if( (sts=blk_out_x( &len, 4 ))) {           /* 転送データ長*/
409:     ;
410:   }
411:   else {
412:     len = req->dmalen * (long)_byte[req->unitcd];
413:
414:     while( len > 0 ) {
415:       wk = len;
416:       if( wk > 16384L ) {
417:         wk = 16384L;
418:       }
419:
420:
421:       wk1 = wk / (long)_byte[req->unitcd];
422:
423:       DISKRED( (unsigned char*)_rw_buf, req->unitcd+_drv, rec, wk1 );
424:
425:       if( (sts=blk_out( _rw_buf, wk ))) {  /* データ長送信  */
426:         break;
427:       }
428:
429:       rec += wk1;
430:       len -= 16384L;
431:     }
432:   }
433:
434:   return( sts );
435: }
436:
437:
438:
439:
440:
441: /*****************************************************************
442:   r_dskotv  出力&ベリファイ処理(?)
443:   ASMを強引にハンドディスコンパイル
444: *****************************************************************/
445: int r_dskotv( req )
446: struct REQ_RW *req;
447: {
448:   return( r_dskout( req ) );
449: }
450:
451:
452:
453: /*****************************************************************
454:   r_dskout  ディスク出力処理(?)
455: *****************************************************************/
456: int r_dskout( req )
457: struct REQ_RW *req;
458: {
459:   int   sts;
460:   long  len;
461:   long  rec;
462:   long  wk;
463:
464:   req->errlow = 0;             /* 下位バイトエラーコード格納  */
465:   req->errhigh = 0;            /* 上位バイトエラーコード格納  */
466:
467:   XCHG4( &(req->dmalen) );              /* もしDOSなら配列変換     */
468:   XCHG4( &(req->starec) );              /* もしDOSなら配列変換     */
469:   rec =req->starec;
470:   rec >>= 16;
471:
472:   if( (sts=blk_in_x( &len, 4 ))) {        /* データ長受信 */
473:     ;
474:   }
475:   else {
476:     while( len > 0 ) {
477:       wk = len;
478:       if( wk > 16384L ) {
479:         wk = 16384L;
480:       }
481:
482:       if( (sts=blk_in( _rw_buf, wk ))) {  /* データ長送信  */
483:         break;
484:       }
485:
486:       wk /= (long)_byte[req->unitcd];
487:       DISKWRT( (unsigned char*)_rw_buf, req->unitcd+_drv, rec, wk );
488:
489:       _rs_buf_clr();
490:       OUT232C('A');
491:
492:       rec += 16384L / (long)_byte[req->unitcd];
493:       len -= 16384L;
494:     }
495:
496:
497:     if( len<=0 ) {
498:       sts=blk_out( &(req->errlow), 2 );
499:     }
500:   }
501:
502:   return( sts );
503: }
504:
505:
506:
507:
508: /*****************************************************************
509:   r_dskctrl  ディスクコントロール
510: *****************************************************************/
511: int r_dskctrl( req )
512: struct REQ_CTRL *req;
513: {
514:   int   sts;
515:   int   n;
516:   long  mode;
517:
518:
519:   req->errlow = 0;             /* 下位バイトエラーコード格納  */
520:   req->errhigh = 0;            /* 上位バイトエラーコード格納  */
521:
522:
523:   n = (int)(req->reqlen);
524:
525:   mode = 0L;
526:   mode = DRVCTRL( mode, (int)(req->unitcd)+_drv );
527:   req->getdat = (char)mode;
528:
529:
530:   if( (sts=blk_out( req, n ))) {    /* リクエストヘッダ送信  */
531:     ;
532:   }
533:   else {
534:     sts = 0;
535:   }
536:
537:   return( sts );
538: }
539:
540:
541: struct  BPB_TBL  {
542:   unsigned short  b_no;          /* セクタあたりのバイト数 */
543:   unsigned char   sct_no;        /* クラスタあたりのセクタ数*/
544:   unsigned char   fat_no;        /* ファット領域の個数 */
545:   unsigned short  rsv_sct_no;    /* 予約領域のセクタ数 */
546:   unsigned short  root_ent_no;   /* ルートの最大ファイル数 */
547:   unsigned short  sct_max;       /* 全セクター数 */
548:   unsigned char   id;            /* メディアバイト */
549:   unsigned char   fat_sct_no;    /* 1fatのセクタ数 */
550: };
551:
552: /*****************************************************************
553:   dskini  ディスク初期化ルーチン
554: *****************************************************************/
555: int r_dskini( req )
556: struct REQ_INI *req;
557: {
558:   int  sts;
559:   struct  DPBPTR  d;
560:   struct  BPB_TBL  bpb_tbl;
561:   int  fat;
562:   int  fat_no;
563:   int  rsv_sct;
564:   char  dsk_flg;
565:   int  drv;
566:   char  d_no;
567:
568:   drv = _drv;
569:
570:   while( 1 ) {
571:     sts = GETDPB( drv, &d );
572:     dsk_flg = 0;
573:
574:     if( sts ) {
575:       dsk_flg = -1;
576:       sts = blk_out( &dsk_flg, sizeof(dsk_flg) );
```

▶ショック! 「おじさん」と呼ばれてしまった。僕は実際の歳よりも若く見えるのに……。 大島 大介(18)北海道

```
577:         break;
578:       }
579:       else if( (sts=blk_out( &dsk_flg, sizeof(dsk_flg) ))) { /*bpb tbl送信*/
580:         break;
581:       }
582:       else {
583:         bpb_tbl.b_no = d.byte;       /* セクタあたりのバイト数 */
584:         bpb_tbl.sct_no = d.sec + 1; /* クラスタあたりのセクタ数*/
585:         bpb_tbl.fat_no = d.fatcount;/* ファット領域の個数     */
586: 
587:         bpb_tbl.rsv_sct_no = d.fatsec;       /*予約領域のセクタ数*/
588:         bpb_tbl.root_ent_no = d.dircount;  /*ルートの最大ファイル数*/
589:         bpb_tbl.sct_max = d.maxfat * (d.sec+1);  /* 全セクタ数 */
590:         bpb_tbl.id = d.id;                  /*メディアバイト*/
591:         bpb_tbl.fat_sct_no = d.fatlen;  /* 1fatのセクタ数*/
592: 
593: 
594:         XCHG2( &(bpb_tbl.b_no) );          /* もしDOSなら配列変換*/
595:         XCHG2( &(bpb_tbl.rsv_sct_no) ); /* もしDOSなら配列変換*/
596:         XCHG2( &(bpb_tbl.root_ent_no) );/* もしDOSなら配列変換*/
597:         XCHG2( &(bpb_tbl.sct_max) );     /* もしDOSなら配列変換*/
598: 
599: 
600:         if( (sts=blk_out( &bpb_tbl, sizeof( bpb_tbl ) ))) { /*bpbtbl送信*/
601:           break:
602:         }
603: 
604:         _byte[drv-_drv] = d.byte;
605:         d_no = 'A' + (char)drv;
606:         d_no --;
607:         printf( "%c: を主機へ仮想ドライブとして登録 ¥n", d_no  );
608: 
609:         drv ++;
610:       }
611:   }
612: 
613:   return( sts );
614: }
615: 
616: /*******************************************************
617:   PC9801機種依存関数
618: *******************************************************/
619: #if SYSTEM==PC98
620: 
621: char      __dmy[128];
622: /***************************************************************
623:   _rs_inz    初期化
624:   return  :  sts
625: ***************************************************************/
626: int      _rs_inz()
627: {
628:   union  REGS  i_reg;
629:   union   REGS    o_reg;
630:   struct  SREGS   s_reg;
631:   int  sts;
632: 
633:   segread( &s_reg );
634: 
635:   i_reg.h.ah = 0;
636:   i_reg.h.al = 0x07;
637:   i_reg.h.ch = 0x4e;
638:   i_reg.h.cl = 0x37;
639:   s_reg.es = s_reg.ds;
640:   i_reg.x.di = (int)__dmy;
641:   i_reg.x.dx = 128;
642:   i_reg.h.bh = 0x2;
643:   i_reg.h.bl = 0x1e;
644: 
645:   int86x( 0x19, &i_reg, &o_reg, &s_reg );
646:   return( i_reg.h.ah );
647: }
648: 
649: int dlytime()
650: {
651:   long  i;
652:   long  a;
653:   long  b;
654: 
655:   for( i=1; i<30000 ; i++ ) {
656:     a=100;
657:     b=200;
658:     a=a*b;
659:   }
660: }
661: 
662: /***************************************************************
663:   LOF232C  受信バッファチェックルーチン
664: ***************************************************************/
665: int  LOF232C()
666: {
667:   union  REGS  i_reg;
668:   union   REGS    o_reg;
669:   int  sts;
670: 
671:   i_reg.h.ah = 2;
672:   i_reg.h.al = 0;
673:   int86( 0x19, &i_reg, &o_reg );
674: 
675:   sts = o_reg.x.cx;
676:   return( sts );
677: }
678: 
679: /***************************************************************
680:   INP232C  1文字受信ルーチン
681: ***************************************************************/
682: int  INP232C()
683: {
684:   union  REGS  i_reg;
685:   union REGS    o_reg,
686:   unsigned int  sts;
687: 
688:   i_reg.h.ah = 4;
689:   i_reg.h.al = 0;
690:   int86( 0x19, &i_reg, &o_reg );
691: 
692:   sts = o_reg.h.ch;
693:   return( (int)sts );
694: }
695: 
696: /***************************************************************
697:   OUT232C  1文字送信ルーチン
698: ***************************************************************/
699: void  OUT232C( int c )
700: {
701:   union  REGS  i_reg;
702:   union REGS    o_reg;
703:   unsigned int  sts;
704: 
705:   i_reg.h.ah = 3;
706:   i_reg.h.al = (unsigned char)(c & 0xff);
707:   int86( 0x19, &i_reg, &o_reg );
708: }
709: 
710: /***************************************************************
711:   GETDPB  DPBテーブル取得ルーチン
712: ***************************************************************/
713: int      GETDPB( int drv, struct DPBPTR  *d )
714: {
715:   union  REGS   i_reg;
716:   union  REGS   o_reg;
717:   struct SREGS  s_reg;
718:   unsigned int  sts;
719:   unsigned int  ds;
720: 
721:   segread( &s_reg );
722:   ds = s_reg.ds;
723: 
724:   i_reg.h.ah = 0x32;
725:   i_reg.h.dl = drv;
726:   int86x( 0x21, &i_reg, &o_reg, &s_reg );
727:   movedata( s_reg.ds, o_reg.x.bx, ds, d, sizeof(struct DPBPTR) );
728:   sts = o_reg.h.al;
729: 
730:   return( sts );
731: }
732: 
733: 
734: /***************************************************************
735:   DISKRED  ディスクリードルーチン
736: ***************************************************************/
737: void DISKRED( unsigned char* buf, int drv, long rec, long len )
738: {
739:   union  REGS  i_reg;
740:   union REGS    o_reg;
741:   struct  SREGS   s_reg;
742:   unsigned int  sts;
743:   unsigned int  ds;
744: 
745:   drv --;
746:   segread( &s_reg );
747:   ds = s_reg.ds;
748: 
749:   i_reg.h.al = drv;
750:   i_reg.x.bx = (unsigned int)buf:
751:   i_reg.x.cx = len;
752:   i_reg.x.dx = rec;
753: 
754:   int86x( 0x25, &i_reg, &o_reg, &s_reg );
755:   dlytime();
756: 
757:   sts = o_reg.x.ax;
758: }
759: 
760: /***************************************************************
761:   DISKWRT  ディスクライトルーチン
762: ***************************************************************/
763: void DISKWRT( unsigned char* buf, int drv, long rec, long len )
764: {
765:   union  REGS  i_reg;
766:   union REGS    o_reg;
767:   struct  SREGS   s_reg;
768:   unsigned int  sts;
769:   unsigned int  ds;
770: 
771:   drv --;
772:   segread( &s_reg );
773:   ds = s_reg.ds;
774: 
775:   i_reg.h.al = drv;
776:   i_reg.x.bx = (unsigned int)buf;
777:   i_reg.x.cx = len;
778:   i_reg.x.dx = rec;
779: 
780:   int86x( 0x26, &i_reg, &o_reg, &s_reg );
781:   dlytime();
782:   sts = o_reg.x.ax;
783: 
784: 
785: }
786: 
787: /***************************************************************
788:   DRVCTRL  ドライブ情報取得ルーチン(ダミー)
789: ***************************************************************/
790: long DRVCTRL( long mode, int drv )
791: {
792:   return( 0x02 );
793: }
794: 
795: 
796: /***************************************************************
797:   blk_in_x  ブロック受信、受信データを並べ替え
798: ***************************************************************/
799: int blk_in_x( unsigned char *data, int len )
800: {
801:   int  sts;
802: 
803:   sts = blk_in( data, len );
804:   xchg4( data );
805: }
806: 
807: 
808: /***************************************************************
809:   blk_out_x  ブロック送信、送信データを並べ替え
810: ***************************************************************/
811: int blk_out_x( unsigned char *data, int len )
812: {
```

▶自分がいつも買う本屋にOh!Xがなかったからあせった。　小口　一法(19)東京都

```
813:   int  sts;
814:   char  wk[4];
815:
816:
817:   wk[3] = data[0];
818:   wk[2] = data[1];
819:   wk[1] = data[2];
820:   wk[0] = data[3];
821:
822:   sts = blk_out( wk, len );
823:
824: }
825:
826: /*****************************************************************
827:   xchg2  2バイトデータの並べ替え
828: *****************************************************************/
829: void xchg2( char *data )
830: {
831:   char  wk[2];
832:
833:   wk[0] = data[1];
834:   wk[1] = data[0];
835:
836:   data[0] = wk[0];
837:   data[1] = wk[1];
838: }
839:
840: /*****************************************************************
841:   xchg4  4バイトデータの並べ替え
842: *****************************************************************/
843: void xchg4( char *data )
844: {
845:   char  wk[4];
846:
847:   wk[0] = data[3];
848:   wk[1] = data[2];
849:   wk[2] = data[1];
850:   wk[3] = data[0];
851:
852:   data[0] = wk[0];
853:   data[1] = wk[1];
854:   data[2] = wk[2];
855:   data[3] = wk[3];
856: }
857:
858: #endif
```

リスト3

```
===============  D3.C  ==================
  1: #define          X68     0
  2: #define          PC98    1
  3: #define          IBM     2
  4: #define          SYSTEM  1
  5:
  6: #if SYSTEM==X68
  7: #include  <doslib.h>
  8: #include  <stdio.h>
  9: #include  <time.h>
 10: #endif
 11:
 12: #if SYSTEM==PC98 || SYSTEM==IBM
 13: #include          <stdlib.h>
 14: #include          <dos.h>
 15: #include          <sys¥types.h>
 16: #include          <sys¥stat.h>
 17: #define           NULL    0
 18: #endif
 19:
 20:
 21: #define  TIMEOUT  5L
 22:
 23:
 24: int  blk_in();
 25: int  blk_inl();
 26: int  blk_out();
 27: int  blk_outl();
 28: void   _rs_buf_clr();
 29:
 30: /*****************************************************************
 31:   blk_in  複数データを受信
 32:   受信に失敗したら5回までやりなおす
 33: *****************************************************************/
 34: int  blk_in( data, len )
 35: unsigned char  *data;     /* 転送データ格納アドレス  */
 36: int  len;        /* 転送データ長        */
 37: {
 38:   int  sts;
 39:   int  i;
 40:
 41:   for( i=0 ; i<5 ; i++ ) {
 42:     sts = blk_inl( data, len );
 43:     if( sts==0 ) {
 44:       break;
 45:     }
 46:   }
 47:
 48:   return( sts );
 49: }
 50:
 51:
 52: /*****************************************************************
 53:   blk_inl  複数データ受信
 54: *****************************************************************/
 55: int  blk_inl( data, len )
 56: unsigned char  *data;    /* 転送データ格納アドレス  */
 57: int  len;       /* 転送データ長        */
 58: {
 59:   time_t  tm;
 60:   time_t  tml;
 61:   time_t  tmx;
 62:   int  bsc;
 63:   int  i;
 64:   unsigned char  *ptr;
 65:   int  c;
 66:   int  sts;
 67:   int  n;
 68:
 69:   sts = -1;
 70:   bsc = 0;
 71:   ptr = (unsigned char *)data;
 72:
 73:   for( i=0 ; i<len ; i++ ) {
 74:     c = INP232C();
 75:     *ptr++ = c;
 76:     bsc ^= c;
 77:   }
 78:
 79:   c = INP232C();
 80:   _rs_buf_clr();
 81:
 82:   if( c!=bsc ) {
 83:     OUT232C( 1 );
 84:   }
 85:   else {
 86:     OUT232C( 0 );
 87:     sts = 0;
 88:   }
 89:
 90:   return( sts );
 91: }
 92:
 93: /*****************************************************************
 94:   blk_out  複数データを送信
 95:   送信に失敗したら5回までやりなおす
 96: *****************************************************************/
 97: int  blk_out( data, len )
 98: unsigned char  *data;     /* 転送データ格納アドレス  */
 99: int  len;        /* 転送データ長        */
100: {
101:   int  sts;
102:   int  i;
103:
104:   for( i=0 ; i<5 ; i++ ) {
105:     _rs_buf_clr();                     /* 通信バッファクリア   */
106:     sts = blk_outl( data, len );
107:     if( sts==0 ) {
108:       break;
109:     }
110:   }
111:
112:   return( sts );
113: }
114:
115:
116: /*****************************************************************
117:   blk_outl  複数データ送信
118: *****************************************************************/
119: int  blk_outl( data, len )
120: unsigned char  *data;    /* 転送データ格納アドレス  */
121: int  len;       /* 転送データ長        */
122: {
123:   time_t  tm;
124:   time_t  tml;
125:   time_t  tmx;
126:   int  bsc;
127:   int  i;
128:   unsigned char  *ptr;
129:   int  c;
130:   int  sts;
131:   int  n;
132:
133:   sts = -1;
134:   bsc = 0;
135:   ptr = (unsigned char*)data;
136:
137:   for( i=0 ; i<len ; i++ ) {
138:     if( LOF232C() ) {     /* 同調機構、受信側でとりこぼし  */
139:       return(sts);     /* 強制リターン        */
140:     }
141:
142:     c = *ptr;
143:     OUT232C( c );
144:     bsc ^= c;
145:     ptr ++;
146:   }
147:
148:
149:   OUT232C( bsc );
150:   tm = time( NULL );
151:
152:   while(1) {
153:     tml = time( NULL );    /* TimeOut チェック */
154:     tmx = tml - tm;
155:     if( tmx> TIMEOUT ) {
156:       break;
157:     }
158:
159:     if( LOF232C() ) {
160:       c = INP232C();
161:       if( c==0 ) {
162:         sts = 0;
163:       }
164:
165:       break;
166:     }
167:   }
168:
169:   return( sts );
170: }
171:
172: /*****************************************************************
173:   _rs_buf_clr  通信バッファクリア
174: *****************************************************************/
175: void _rs_buf_clr()
176: {
177:   int  c;
178:
179:   while( LOF232C() ) {
180:     c = INP232C();
181:   }
182: }
```

▶「あすか120%」の記事を呼んだところ，システムに非常に興味をもったのですが，「女の子ゲーム」「よくしゃべる」この2点でとても買えません(周囲の目が怖い)。このシステムを受け継いだ次回作を待つことにしましょう。ところで，誰か「マッドストーカーX68」の隠し技を教えてください。　小野　孝平(17)東京都

愛読者プレゼント

1 大魔界村

3名

X68000用

5"2HD版　9,800円(税別)

美しいグラフィックに超ムズのゲーム。これがマニアにはたまらない快感!?　でも画面に見とれる余裕は……。とにかく，腕を磨くのだ！　Oh!X編集部内でも，攻略に写真撮影にと苦労した逸品です。

▲カプコン

2 F-Calc for x68k

2名

X68000用

3.5/5"2HD版　14,800円(税別)

TAKERU　14,800円(税込)

6月号で紹介した表計算ソフト。操作系などはあの「Lotus1-2-3」に似ています。製品には，5,000円でプログラムソースが入手できる申し込みはがきもついていて，自分なりに改造もできちゃいます。

▲クレスト　☎03(3418)5993

3 Z-MUSIC システム ver.2.0

5名

4,800円(税込)

今月号は，特集がコンピュータミュージックということで，X68000ユーザー必携(?)のこの1冊をプレゼント。まだ入手してないそこのキミ，出遅れてるぞ。待ちきれない人やハズレた人は，書店に走るのだ！

▲ソフトバンク　☎03(5642)8101

4 CD「No Frame No Fame /X'Mass Song 68K」

1,000円(送料・税込)

50名

Musicstudio発売5周年特別企画として作成されたCDです。Mu-1Superユーザーのバンド「星智輝with T.T.」の作品で，第1回シャープ芸術祭全国大会でMusic部門賞とRoland賞を受賞したものです。

▲サンワード　☎044(855)4335

5 「世界最大の恐竜博」入場券

3組6名

6月18日～9月4日に大阪で開催される「世界最大の恐竜博」。その会場内で開催されるテーマパーク「DINO・PARK」を企画・運営のヒューマン(株)より，入場券をプレゼント。イベント内容については，ペンギン情報コーナーをご覧ください。なお，開催期日の都合上，このプレゼントはチケットの発送をもって当選者の発表に代えさせていただきます。

▲ヒューマン　☎0422(70)7777

プレゼントの応募方法

とじ込みのアンケートはがきの該当項目をすべてご記入のうえ，希望するプレゼント番号をはがき右下のスペースにひとつ記入してお申し込みください。締め切りは1994年7月18日の到着分までとします。当選者の発表は1994年9月号で行います。また，雑誌公正競争規約の定めにより，当選された方はこの号のほかの懸賞には当選できない場合がありますので，ご了承ください。

5月号プレゼント当選者

1マルチメディアスピーカーEAB401　(京都府)山地将朗　(愛媛県)坂本和秀　2卒業～GRADUATION　(北海道)水戸部広明　(埼玉県)日下高志　(東京都)生井　務　3DoubleBookin'　(神奈川県)安沢光男　(奈良県)今井健生　(香川県)羽原健司　4デジタルアートコレクション vol.9　(新潟県)舘　秀明　(東京都)山口　茂　(奈良県)開口嘉雄　(大阪府)山川勝也　(鹿児島県)大山茂樹　5マウスパッド　(青森県)新谷貴幸　(埼玉県)松本吉広　(静岡県)鈴木善男　(長野県)髙橋直樹　(愛知県)平松一成

(敬称略)

以上の方々が当選しました。商品は順次発送いたしますが，入荷状況などにより遅れる場合もあります。

「新しい環境への順応」に関する起承転結

[起] 無数のプログラミング言語の存在意義

「Catalog of compilers, interpreters, and other language tools」という名前のついた記事が，またネットワークを流れてきました。この記事には，プログラミング言語(ツールなども多少含まれる)が山のように紹介されています。ざっと調べたところ，400近くもありました。

400という数字は一見,大きく見えるかもしれませんが，実はこの記事で紹介されているプログラミング言語などは，氷山の一角にすぎません。なぜかといえば，ここに挙げられているのは，ソースコードまで公開されており，ネットワーク経由で自由にもっていって自分の計算機の上でコンパイルし直して結構ですよ，というものだけだからです。

自由に使ってくださいとネットワークで公開されているプログラミング言語は，この世に存在するプログラミング言語全体からみれば，ほとんど誤差のうちといってもいいほどのわずかなものでしょうから，この世に存在するプログラミング言語の数がいくつあるかなどということを，この数字から推定するのは，しょせん意味のないことかもしれませんね。

要するにプログラミング言語の数は無限大といっておけばよいのです。それでは，なぜ，無数のプログラミング言語が存在してしまったのかといえば，次に挙げるとおりでしょう(この部分は「情報科学」という講義の，最近行った分のメモからの引用です)。

1)「何を記述するのか，何を計算したいのか」という目的ごとに，最適なプログラミング言語は異なる

2)「プログラミングをどう行ったら効率が上がるか」というプログラミング方法論自体が発展途上であり，特長のある方法論が生まれるたびに，それに即した新しいプログラミング言語が作られる

3) そもそも人間には千差万別の好みがあるので，無数のプログラミング言語が必要である

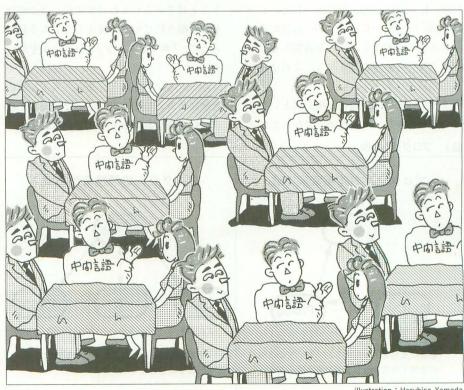
しかし，現在でこそ無数にあるわけですが，プログラミング言語の数が1あるいは0に収束するという状態がひとつの理想的な姿として考えられます。たとえば，日常しゃべったり書いたりしている自然言語をベースにして，それから無駄な部分を省き，あいまいな部分をそれに補足すると，究極的にはただ1つのプログラミング言語が考えられるでしょう。

また，計算機の概念から，現在我々が考えているような意味でのハードウェアとソフトウェアの区別がもはや存在しないような知的なマシン，つまり，知能機械が生まれたならば，そこにはプログラミング言語なる概念はもはや存在しないといえましょう。

しかし，遠い夢物語ではなく，もっと現実的に考えるのならば，プログラミング言語がこのように無数にあることは，3つの理由を先に挙げたように，好ましいこと，あるいは必要であることには間違いがないと思います。

しかし，無数にプログラミング言語があると，あるプログラムを別の処理系にもっていって同じように動作させることや，そのプログラムの意味を理解することが難しくなってしまいます。このことは，プログラムの寿命がきわめて短くなることに直結しており，非常に重要な問題です。

[承] 移植のオーバーヘッド

そのような問題を解決するための率直なアプローチのひとつとして，あるプログラミング言語の規則に基づいて書かれたプログラムを，その意味は変えないままで別のプログラミング言語の規則に基づいて書いたように,自動的に変えてしまうという「プログラム変換」の研究が活発になされています。

illustration：Haruhisa Yamada

「新しい環境への順応」に関する起承転結

似たような文法をもつプログラミング言語間で変換を行うのならば，自動化もそう難しくはなく，機械的にパターンを見つけて，それを置き換えていくだけで成功しそうに思えますが，言語間にへだたりがあればあるほど，そのプログラムのもつ「意味」やアルゴリズムといった抽象的な段階まで計算機が理解しなければ，変換することはできなくなっていくわけです。

プログラムのもつ「意味」がどこまでわかるかということは，結局，「意味」というものをどこまで定型化できるかということにつながるわけで，それは「意味」自体の意味にも関わる本質的な問題で，そこには困難がいくらでもころがっていそうな感じです。

ところで，言語がn個あって，それらの間でプログラムを直接変換するためには，n×(n－1)個の方式を確立しなければなりません。たとえば，LispとCとPrologの間で変換する方式は，nが3ですから変換方式は6個必要となります(Lisp→C，C→Lisp，Lisp→Prolog，Prolog→Lisp，C→Prolog，Prolog→C)。つまり，言語が1000個あったら，100万もの方式を研究しなければならないというのですから，これは非現実的な話といえます。

そうではなく，共通の表現形式というものを定めておき，ある言語で書かれたプログラムをいったんその表現に直してから，それから目的の言語に変換するというように，1クッション置く方式にするのならば，組み合わせの数はぐっと減って，n×2でいいことになります。

nが3ならば，たまたま先の方式の数と同じ6個です(Lisp→中間言語，中間言語→Lisp，Prolog→中間言語，中間言語→Prolog，C→中間言語，中間言語→C)が，nが大きくなると，その差は歴然とします。nが1000になっても，方式数は2000でいいわけですから。

ところで，プログラムを移植するという観点は，プログラムというものを，受動的で知的な要素がまったくないものとみなしていることを意味しています。ここで，見方を変えて，プログラムが自分で環境に合わせて変わるということを考えます。プログラム自身に，もう少し知的になってもらうわけです。

プログラムという概念，呼び方でも，自分で自分の動作を状況に応じて変えるということは考えられるわけですが，ここではもっと根本的に知的に脱皮してもらう結果として，「エージェント」という名前を使います。

エージェントという言葉は，人工知能研究をはじめとして，いろいろな場面でそれぞれに適当な意味をもたせて使われます。

とりあえずここではシンプルに，「計算機の中で自律的，分散的に動作するひとつの主体/ソフトウェア単位」と定義してしまいましょう。

それをとりまく環境が変わったとしても，別のプログラムの指示があるわけでもなく，人間が手で変更するのでもなく，「自律的」そして「分散的」に，その環境に適応して自分自身の動作を変えることができるという知的な主体，それがこの文脈におけるエージェントなわけです。

プログラムの変換方法を言語に応じて次々と考えていくという方法も，それはひとつの有力で現実的な方法ですが，プログラムというものを，より主体的でやわらかいものへと変質させていくということは，新しいハードウェアやソフトウェアの枠組みを想定するきっかけになるのではないかと思われます。

[転] 連続してふりかかるアクシデント

そうはいうものの，「エージェントとはいったい何なのか，結局よくわからない」と思われる方がほとんどかもしれません。エージェントのひとつの完成された姿，それはもちろん計算機の中ではなく，ここにいる我々自身です。

我々は，マインドコントロール(すでに懐かしい言葉だ！)されているのでなければ，自律的に分散的に動作している能動的な主体であります。

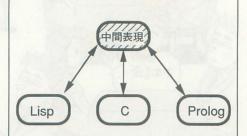
そして，そのような存在であるからこそ，環境が変わるという負荷にも，自分でその動作の仕方を変えて対処することができるのです，できるはずなのです。

具体的な例(インスタンス)として，この僕自身がいます。この4月に大学を転任し，住む場所も，20年ローンのおかげもあって変わりました。そして，この移植というか，変換というか，その負荷にも，自分を変えることによって，軽々と耐えられたはずなのですが……。

図1 プログラム変換

プログラムの直接変換方成

Lisp
C
Prolog

新しい言語を1つ加えると，すべてのほかの言語との変換手法を考える必要がある

新しい言語を1つ加えても，中間表現との変換手法を考えるだけでよい

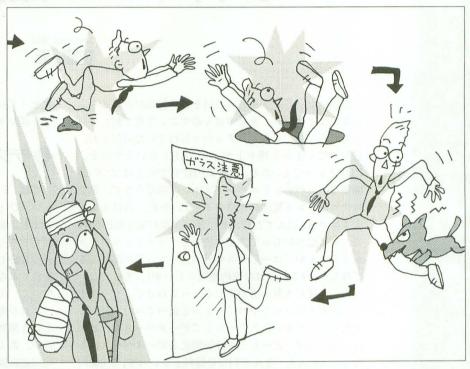
ところが，これが，はっきりいってまったくだめで，エージェントどころではありません。ただただもう予期せぬアクシデントの連続で，この数カ月，知能機械どころか，まるで低能機械な生活を送り続けています。そして，知能機械としてやらなければならないことの山は増える一方でして，残念なことに最近では，次にやるべき優先度の高いものを選ぶことにさえ，時間がかかるようになってきました。

新しい環境に順応どころか，移る作業そのもののなかでも，いろいろと思いがけないことが起きてしまいました。まず，研究室内の本などの移動のときに，本をしばった束を1つ，どうも荷台から落としてしまったようなのです(自分たちで引っ越しをしたので，泣きつく相手もいません)。

しかも，その束というのが，この1年こつこつと自腹を切って買い集めた，大事な人工生命に関する国際会議の論文集なのです。この損害は，10万円は下らないでしょうし，いろいろと書き込みをしていたので，値段ではいい表せません。

おまけに，家のほうの引っ越しでも，とんだ災難に遭いました。大きな本棚のガラスを2枚割られたのです。しかも，引っ越し業者は，「たかがガラスの1枚2枚」「お客さんが割った可能性もある」「けんかだから両方のいい分を聞かないと」などと，こちらの被害者のほうをまるで悪者扱いするのです。

また，新しい大学のほうも，文化の違いがあって，それに慣れていないからでしょう，何をするにも10倍の時間やエネルギーが必要です。しかし，その一因は僕自身にもあると思っています。僕と接するほぼ全員が，僕のことを学生とみなして疑いもしないのでして，事務室で話をしようとしても，「(その仕事を頼んだ)先生の名前は？」とか聞いてくれるならまだましで，おちおちしていると，「ちょっと。邪魔っ」という感じではねとばされます。

さらに作り話のようなのですが，このあ

いだの土曜日，道路を歩いていて，落ちていた1枚のトランプのカードを拾うと，それがスペードのエースでして，これが人の話によれば，えらく縁起が悪いそうで，実際，その2日後に考えられないような大ポカを演じてしまいました。

ほかにも銀行のミスなのにこちらが10万円とられそうになったりだとか，とにかくグチばかりいいたくなるようなことが，職場あるいは住居を移ることによって連続して起こってしまい，「いったい，なぜここまで，環境の変化によって大きなダメージを受けねばならないのだろうか」と考え込んでしまったのでした。

[結]
知能の本質へ

我々は，プログラムのように新しい環境のなかでもきちんと動くように誰かに変えてもらうのをただ待つのではなく，自分で自分を変えることのできる超優秀な自律分散エージェントです。そして，自分を自分で変えるということはきわめて重要なことです。

赤子は，ほったらかしにしておくと死んでしまいます。なぜそんなにも未完成なまま生まれてしまうのでしょうか？　遺伝子の中にもっと重要な情報を残して，最初から大人並みにして生まれるほうがよいようにも思われます。

しかし，実は，我々のもつ知能の本質はスタート地点から始まっているのです。つまり，完成品をどんと作り出すのではなく，どんな場所に生まれてもその環境に合うように，未完成の部分を人の助けを借りながらも自分で作り上げていくのです。つまり，わざと遺伝情報を中途半端にしておいたというわけです。

現在の僕のように，新しい環境に適応しようとする苦しみ，これこそが知能の本領発揮の場面なのでしょう。お定まりの日常の繰り返しをしていたのならば，知能も腐ってしまうというものです。とにかくとにかく，このどん底の状態こそ，実はすごく喜ばしい状態なのだーっ！

(……でも，やっぱり，せめてひと月だけ時間さん，止まってくれませんか？　何とか遅れを取り戻しますから)

石の言葉，言葉の夢

[第4回]

ワイドテレビの胡散臭さよ

Ogikiubo Kei
荻窪 圭

以前，コードレスモデムの話を書いたのだけれども，なんのことはない，世の中はものすごい速度で動いているのであった。くそぅ。すっかり忘れていたのだよね。1992年にちょっと盛り上がったPHPとかいうやつを。PHPといっても，あの，松下系PHP研究所とは(たぶん)関係ない。パーソナルハンディホンの略だそうで，第2世代コードレス電話の規格のこと。いやあ，『日経エレクトロニクス』読んでたら，松下通信工業が家庭用PHP発売，って記事があって，よく読むと「コードレスファクシミリ」とか「コードレスモデム」って言葉が出てくるではないか(これらは将来的機能らしいが)。参ったな。

参ってばかりじゃ男がすたる，ってわけで，パーソナルハンディホンってやつ，ちょっと調べてみた。調べるったって，ネットにアクセスしてデータベースサービス行って，「くそ，どーして日経新聞記事データベースはこんなに高いんだ。なんで利用登録料とかいって，5,000円もとりやがるんだ。あほか」ってひとしきり怒ったあと，利用登録料のいらない(もちろん，使用料はけっこうとられるが)日刊工業新聞記事データベースを呼び出しただけだけど。

それにしても，なんでG-Searchっていうデータベースシステムは使いにくいのだろうね。

何がひどいって，検索結果を画面に出力するではないか。その中から，読みたい記事をピックアップして表示させる。内容にもよるが，記事の見出しだけではわからんので，関係ありそうなものを指定する。たいてい，数十件ある。記事を番号で指定して，えいや，と，リターンキーを押す。だらだらと記事が垂れ流される。それだけならまだいいが。

これがまた，1件1件出力するごとに「まだ続きがあります」って，改行キーを押せというのだ。そんなもん，一気に全件出力してくれよぉ。何十件もある記事を，1件1件改行キー押すなんざ，正気の沙汰でない。どうせ，全部ダウンしてから読むのだから，無駄にマシンの前に座っていなければならんし，ダウンをバックグラウンドにまわしてその間にほかのお仕事を，ってのもできないのだ。困ったものだ。そりゃあ，通信ソフトでマクロ書けばなんとかなるが，そういう問題ではないと思う。

で，何十回と改行キーを押していると，出てくる出てくる。いちばん古いのが，1992年。思い出すなぁ。そういえば，コードレスホンの子機が携帯電話みたいに使える，ってノリの記事を読んだ記憶がある。1993年の実用化をメドに，ってあるけど，いまはもう1994年。まあ，1年ばかり遅れて，松下が製品化したわけだ。

デジタルなパーソナルハンディホン

では，PHPとは何か。どうやら，2つの売りがあるらしい。ひとつは，親機を介さずに，子機同士で通話できること。親機を通さない→子機が2台あればどこでも通話できるってことで，要するに，全2重のトランシーバーって感じだ。しかも，親機を通さない→公衆回線網に接続されない，わけだ。出かけるときに子機を2台持っていく，というのも変な話だけれども。

これが発展すると，子機を携帯電話のように使える，という屋外利用の世界が展開される。ただ，携帯電話と違い，あくまでもコードレスホンの子機であるから，利用可能範囲が狭い。それを携帯電話のように使うには，100m単位クラスで中継基地局の設置が必要で，インフラが非常に大変になる。どうなるのかね。具体的にはまだわからんけど(札幌や東京で実験しているそうだけど)。

もうひとつの売りが，デジタル化。PHPはデジタルコードレスホンなわけで，通信速度は14400bpsクラスになりそうということらしい。データ通信用としては，まあ，そこそこの値だ。これで，内蔵PHPモデムなんてものができれば，屋内だけでなく屋外でも使える，というおいしい話になる。いまのデジタル携帯電話用モデム接続ユニットは2400bps固定だそうだけど(とほほ)，まあ，こちらも14400bpsまではいってくれないとなあ。寂しい。

もちろん，携帯電話とPHPは違うから，本格的なモービルコンピューティングには携帯電話でないとつらいけど，ホームコンピューティングという観点からいえば，携帯電話である必然性はないのだよね。

ここで，松下通信工業が発売したPHPに目を戻そう。この製品では(『日経エレクトロニクス』によると)，「親機と子機の間でデータをやりとりするための制御手順を統一方式として決めた」そうである。もう，私が4月号で書いていた話そのまんまだ。

いま，松下が面白い

コードレスホンとテレビとパソコン。というわけで，もうちょっと松下の話。だって，面白いんだもん。

まずは，X1→X68000ときたパソコンテレビ路線が復

活しようとしている。わはは。いつも早すぎるシャープ，ってわけだ。

松下とIBMがディスプレイ一体型のパソコンに標準でTVチューナーをつけた。X1はパソコンのディスプレイとテレビを合体させ，それにパソコンをつないだ。だから，今回のとはちょっと違う。IBM，松下，アップル(アメリカで発売したMacintosh TV)はパソコンが生きてないとテレビも見られないわけだしね。

それでも，パソコンのX1化には変わりない。今回のテレビとの合体は，マルチメディアソフトの流行に便乗した，ってのが正しいところだ。どれも，サンプリング音源，CD-ROMドライブ，ステレオスピーカーを標準搭載している。これで，一体型パソコンを1台買ってくれれば，テレビ，パソコン，CDプレイヤーなどなどがこれ1台で済むわけで，自分だけの空間が欲しい人にはまことありがたい。

これとは逆のアプローチが，東芝のDynaBook EZ Visionで，これは，DynaBookにNTSC出力機能を標準装備したもの。製品としての完成度や値段はともかく，ノートブックパソコンをいまの大画面テレビにつなごうという発想だ。

いずれにせよ，いままでパソコンに縁のない人にも使わせるには，テレビとの合体が不可欠らしい。とにもかくにも，X1的だ。ところで，当のシャープは何をやっているのでしょう(笑)。

松下といえば，3DO以来テレビまわりにかなり力を注いでいるようで，とうとう，テレビ＋ビデオ＋CDラジカセを合体させてしまった。これは笑える。CRTの下にビデオデッキ，上にCDプレイヤーとカセットデッキがくっついているのだ。熱がこもったりしないのかなあと心配してしまうが，まあ，大丈夫なのでしょう。

テレビとパソコン，ってことで考えると，気になるのが，CRTと使う人との距離だ。パソコンは，CRTと使う人の距離が(物理的にも精神的にも)近い。テレビは非常に遠い。ファミコンはその中間くらい。これは，インタラクティブ性を備えたマルチメディアソフトを見る，といったケースで非常に重要な意味をもってくる。単純に考えてはいけない。

この辺の話は来月する。たぶん。

横長テレビについて

巨大化といえば，横長テレビを買おうかと思っている。しかし，どうにもこうにも胡散臭くて踏み切れない。

まず，価格が胡散臭い。横長テレビは高いのである。同じようなサイズの場合，4：3サイズのテレビに比べて，実売価格で倍くらいの違いがある。どうしてこんなに高いのか。

続いて，サイズ表記がようわからん。ブラウン管のサイズって，対角線の長さをインチで表示するから，感覚的にピンとこないのだ。そこで，調べてみると(東芝のカタログにはちゃんと縦横それぞれのサイズ比較表が載っていた。偉い)，横長テレビの24インチっていう，いちばん小さいヤツの高さが，4：3モニタの19インチよりちょっと小さいくらいであった。24インチって，店頭で見ると小さいけど，私の頭の中で，24インチ横長＝19インチ現行，って式が成り立った。よし，24インチにしよう。

横長テレビを買ったって，どうせほとんどは4：3サイズの映像を見るのだから，19インチテレビだと思えばいいのだ。いま私が見ているテレビが15インチだから(いうまでもなく，初代X68000と一緒に買った，年代物のディスプレイテレビである)，まあ，許せる。

続く胡散臭さは，4：3の映像を16：9で見るのはサギみたいなもんではないかということ。上下を切ってトリミングするのも，横に引き伸ばすのも。

これも，いろいろとパターンがあるらしい。横長ブラウン管の両端を使わずに，表示する(19インチテレビと同じ)。横に引き伸ばして表示する(みんな，いろいろ考えるもんで，中央付近では引き伸ばしがなく，両端にいくにつれて伸ばしてワイドに見せるって技が一般的になったらしい)。上下を切って表示する。

サギみたいなもんといえば，シネスコサイズやビスタサイズの映画を無駄なく見られる，っていう話。これは要するに，上下を切って，真ん中を拡大表示しているのと同じだ。パノラマ写真同様，トリミングしているだけだ。

パノラマサイズの写真というのも，実は35mmフィルムの中央部分だけを横長に使う。普通に撮ってトリミングするのとなんら変わらないわけだ。で，プリントするときは，パノラマサイズとかいってでっかく引き伸ばすので，当然ながら画質はどんどん粗くなる。使い捨てカメラで撮ったパノラマ写真なんて，見られたもんじゃないもんね。最初，35mmフィルムを2コマ分使って撮影するのかと思っていた私はぶっとんだ。アイデアといえばアイデアだけど，これ，サギみたいなもんじゃん。パノラマ写真をありがたがる国だから，ワイドテレビも売れるのだよな。普通の人って，こういうの，気にならないのかしら。

横長テレビが普及したら，ワイド対応のハンディビデオカメラが絶対出てくるな。で，パノラマ写真と同じで，実は上下に黒い筋を入れてトリミングしているだけ，っていうの。ああ。

さらに，横長テレビの胡散臭いところといえば，EDTV2。本当に2年後だかに横長テレビの放送が始まるのか。

と，こうして考えると，やはり胡散臭い横長テレビである。規格としては，高画質テレビまでの過渡期的製品であることに間違いないと思うから無視してもいい気がするが，なんか，胡散臭さに惹かれるのだ。だって，面白そうなんだもん。

はてさて，パソコンとは全然関係のない話になってしまったが，レーシングシミュレーションなんかを横長テレビで遊んだらさぞかし気持ちいいだろうなあ，とふっておいて，ごまかそう。

第92回

猫とコンピュータ

はじまりの5秒間

Takazawa Kyoko
高沢 恭子

たいていのひとにとって，いきなりパワー全開，全力疾走で何かをするのは難しいことでしょう。これは，機械も同じこと。準備。下ごしらえ。助走。初期設定。面倒なことはいろいろありますが……。

猫にはかなわないと思うことはたくさんあるけれど，そのなかでもいちばんマネできないのが，朝の立ち上がりの速さ，つまり1日のスタートのすばやさだ。

ホンニャアは，仮眠か熟睡かはわからないが，ともかく寝ている状態からいきなり目的の行動をはじめることができる。

目ざましのオルゴール時計が鳴ると，とび起きる。朝食のために早足でキッチンにいく。朝のパトロールに出かける。ネボケたり，迷ったりしているようすがない。

めんどうなんだよネェ

ホンニャアが爪先にサクラの花びらをつけて帰ってきた。東京のお花見はもう終盤だった。

「そっちのサクラは終わったの？」

三重のマンションにいる夫との電話で，私はたずねた。

「まだまだ，きれいだよ」と夫。

「お掃除はしてるの？」

「ちょっとサボってる」

「ちょっとじゃないでしょ」

「掃除機ってのは，アレ，めんどうだね」

「どうしてェ，カンタンじゃない」

「めんどうだよ，立ち上げるのが。パソコンよりめんどうなんじゃないかなあ」

「えーっ?!　そうかなあ」

「すっごい，めんどうだよ」

なぁるほど，やっぱりめんどうなのかと私は思った。掃除機ではなく，パソコンのほうだ。夫もパソコンの立ち上げは少なからずめんどうだったのだ。

パソコンを使おうとするときの，あのオックウな気持ちはなんなのか，いつもフシギだった。立ち上げは電源を入れて待つだけで，こちらが努力しなければならないことは何ひとつないのに，なぜかめんどうな気がする。このあたりが，ニセモノユーザーの証拠かなとも思っていた。

ところが，パソコンとのつきあいが誰よりも長く，その相性も人一倍と思われる夫が，たしかに，パソコンの立ち上げもめんどうだという発言をした。

＊　＊　＊

数日後，大学に通いはじめたトオルと，マイペースのホンニャアを残して，三重に向かった。トオルに，自分の身のまわりのいっさいを自分でやるたいへんさと楽しさを体験してもらうため，これからは月の半分以上を，ひとりで生活させようということになったのだ。

夫のほうも，勤務のための暮らしとはいえ，3DKのマンションをほとんど全部マシンルームにして，楽しく生活しているらしい。私はこれから当分のあいだ，視察と手入れをかねて毎月2つの住まいを往復しながら，自分も楽しむことにきめた。

三重では時間をみつけては，夫がクルマでサクラの名所につれていってくれた。

東京にいると，ソメイヨシノが咲いて散れば，それがサクラのすべてだったが，ことしは西にやってきて，いままでとはちがった，豊富でおおらかなサクラを見ることができた。サクラにはこんなにも多くの種類と姿があり，咲く場所によって花期も生命力もさまざまで，百の表情をもつものだということも知った。

奈良の吉野山の，稜線をおおうみごとな花の群れ。津風呂(つぶろ)湖畔に沿って流麗に枝を連ねる並木のサクラ。月ヶ瀬に咲いていた静寂な一群。岩倉峡の渓谷を見おろして，のびのびと咲く山桜。

柳生の里，忍辱(にんにく)山円城寺，白毫(はくごう)寺，般若寺。山門に，境内に，池の端に，芽ぶきはじめた周囲の緑との調和が，またみずみずしい。こんなに長期にわたってサクラをながめた春もはじめてだった。

＊　＊　＊

「だいじょうぶ？　朝のサラダはつくってるの？」

東京のトオルに電話した。

ふつうの下宿暮らしとちがうのは，通常の家庭としての設備がすべて整っていること。その点は優雅なものだが，郵便物や宅配便，あらゆる用件の電話や外来者への対応など，ふつうの家庭なみにこなさなくてはならない。なかには税金や金融関係の連絡もある。事実上の世帯主なのだ。

「サラダは毎朝ちゃんと食べてるよ。野菜イタメも開発したからね」

「ほんとォ？」

リンゴの皮もむいたことがないのに，長足の進歩だ。

「料理はおもしろいよ。それより，ホンニャアとの連立のほうがタイヘンだよ」

「でも，お掃除はしないでしょ？」

「うーん，台所や食卓の下は，よくゾウキンがけしてるけどね。掃除機をかける掃除はしてないなあ。持ち出すのがめんどうなんだよねえ」

やっぱり，夫と同じことをいう。

好きならガマンする

夫とトオルのためにすこし弁護しておくと，掃除機は敬遠しているものの，2人の生活環境はそれぞれの方針にもとづいた整理整頓，分類がおこなわれ，それなりにキレイなのである。

キレイの中味を，ものが整理されてかたづいていることと，ホコリや汚れが除かれていることにわけてみると，2人とも前者については積極的につとめている。それは

気持ちがよいこと以上に，自分にとって便利だからである。

これに後者の手段として掃除機をかければ，キレイが万全であるということも知っているが，そこまではしなくてよいだろうと思っているらしい。ホコリと汚れも除いたら気持ちがよいにはちがいないが，さほど自分の便利には直結しない。

ホコリの実害が自分の身にせまってくれば別だが，掃除機などを使う本格的(?)な清掃は妻や母のしごとであると，どこかで思っていることもあるだろう。それに，なんてったって，あの掃除機をかつぎだすのがめんどうくさい。

私はどうして掃除機を使うのだろう。掃除機の立ち上げをヤッカイとは思っていないのだろうか。

私の場合，掃除はおフロと似たようなサッパリ感とケジメの効果があるので，誰のためより，自分のためにしているように思う。気がすむからという理由がいちばん大きいだろう。だからキレイを成り立たせる2つの要素を両方クリアしたいために，掃除機も使う。掃除機はかなり信用できる機械だと思っている。

では持ち出すことがめんどうではないかというと，じつはやっぱりめんどうだ。めんどうで，めんどうでたまらない。それでも，好きなことはやめられない。夫のパソコンと同じだ。

濃密な数十秒

パソコンと掃除機の立ち上げをマジメにくらべてみた。

パソコンの立ち上げはどんなことがおこなわれているのか。

まずスイッチを入れる。これで機械の目がさめる。自分はCPUであると認める。さて周辺機器は何があるだろうと，カッチン，カッチンいいながらたしかめる。キーボードはあるか，CRTはついているか。マウスはあるのか，ないのか。

ハードディスクはあるか，それともフロッピーディスクなのか。モデムや電話コードはつながっているか。

さらにハードディスクかフロッピーディスクの命令内容を読み出していく。バッファはどのくらいか。ファイルがいくつあるか。VJEはあるか。こんなことで，けっこう数十秒かかる。つまり，機械が目ざめて起きあがるまでの時間だ。

ゴールデンウィークに遊びに出かけた先で，知人の3歳になる娘さんが，お昼寝から目ざめたときもこんなふうだった。

はじめは自分が何者なのかもわからないようすで，不きげんにぼんやりと髪の毛をかきむしっていた。ココハドコ？ワタシハダレ？　の状態だ。やがて自分に気づき，周囲の状況をひとつずつたしかめ，ようすがわかったところで，だんだんといつもの活動にうつっていった。

掃除機の立ち上げはどうか。

こちらは電源を入れるまでが立ち上げのしごとで，人間の手と力が必要になる。それとともに，立ち上げの動作をしながら人間の気持ちがいちいち介入する。

まず置いてある場所から機械を運びだす。このときいちばん抵抗を感じる。つぎに分解されていたホースをつなぐ。長すぎるのは使いにくいから，3つのうちひとつは残しておこうなんて考える。

頭部がついていないときは，とりつけなければならない。そのあとコードをいっぱいに引き出すが，ひっぱりすぎると故障につながるので用心がいる。しあげはコンセントにつなぐことで，ここでどの位置のコンセントを選ぶかがたいせつだ。動力の基地として，一度の接続でなるべく広範囲の掃除ができるような，有効な場所につなげなくてはならない。ここまでの所要時間はパソコンとよい勝負だ。

みずからの脳を目ざめさせるパソコンの自動立ち上げと，脳をもたない掃除機に物理的な組み立てをほどこす立ち上げ。便利な機械を使う前のほんのわずかな時間は，どちらもとても濃密だ。

ガマンのしどころ

「掃除機をかけたよ。ボクの部屋のジュウタン，あんまりヒドイから」

トオルが見るに見かねて，とうとう掃除機を持ちだしたらしい。やってみると，いままで見えなかったゴミが見えてくる。ゴミが気になりだすと，掃除は欠かせなくなる。

はじめて自分の意思で掃除をするのはどんな気分だろう，そんな思いもあってあらためて掃除機を使ってみたが，これはやっぱりラクとはいえない。

illustration : Kyoko Takazawa

掃除機が自分のすぐそばにあっても，手元まで運んできて立ち上げるまでのめんどうくささに変わりがない。そして，むしろ電源を入れてからの動作に問題がある。

掃除機というものは，働いているあいだじゅう，なんと力のいるものか。吸引力の強さに耐えるだけの力で，ずっとホースを支えつづけなければならない。これはなかなかの労働だ。しかも掃除をする場所に合わせてパワーの調節をしたり，手の角度を変えたり，アタッチメントを交換したり，いつも対応を考えながら動かさなくてはいけない。

操作中は換気がいること，ものを選ばずに吸い込んでしまうことなど注意項目もあるが，最大のネックはコードでつながれていること。このために制限される動きも多い。もちろん，これらを上回る集塵の威力が衛生面の大きな手助けをしてくれるわけだが，どう考えてもパソコンより力がいるし，めんどうだ。

ものの準備や立ち上げは，どんなことにもある。電話やFAXだって立ち上げがあって，それなりのめんどうもある。クルマはどうだろう。完成度が高いから，あまりわずらわしさがないようにも思う。

ホンニャアにはおよばないが，人間の生活上の立ち上げもたいせつだ。メンタルな立ち上げがうまくいかないと，登校拒否や出社拒否なんて深刻なことも出てくる。

エネルギーがいちばん必要なのは，はじめの数分，それも最初の5秒間がガマンのしどころだ。これを過ぎれば，ひとまずスタートできる。

Another CG world
©KYOKO EGUCHI

んー
なんじゃー?

流れ星にちがいないから
願いごとをすればかなう
かもしれない…って
ホンマかいな?

でも,もし
ホントーなら

宝くじ
1等当たって
おお金
もち
道で大金
ひろって,落とし
主あらわれず
おお金
もち
Jump
に連載で
おお金もち
とにかく
おお金もち
になる
ぎんぎら
ぎん

おお金もちになって……
ビバリーヒルズにプール付きの大邸宅を買う…
やきソバパンお腹いっぱい食べたい
ティファニーの宝石を思いっきり買う
カップラーメン一生分がほしい
欲
一生働かずに楽に暮らす
3倍速CD-ROMドライブがほしい
ケーキ食べほうだいに毎日行きたい
株に投資してひともうけでもっとおお金もち
タイに行って象さんに乗る
欲
欲
8

あーこれだからやだよ…人間は
9

10

ずう〜
ずう〜
宝石
株
金
大邸宅
欲
おお金もち
別荘
11

ずう〜
ひらひら
12

しゅぼ
ころん
13

お星さまがまた空に帰れますように…
14

MAT 28 '94

お〜い出してくれ〜い
シュワッチ
15

PENGUIN INFORMATION CORNER

ペ・ン・ギ・ン・情・報・コ・ー・ナ・ー

NEW PRODUCTS

3.5インチ光磁気ディスク
MO-120G/120S/230S
アイシーエム

MO-120G

MO-120S

アイシーエムは3.5インチ光磁気ディスクSCENAGEシリーズ「MO-120G」とSTRIDEシリーズ「MO-120S」「MO-230S」の3機種を発売する。

主な仕様は以下のとおり。

●MO-120G
記憶容量：121Mバイト
平均シーク速度：120ms
ディスク回転数：1,800rpm
バッファ容量：128Kバイト

●MO-120S
記憶容量：121Mバイト
平均シーク速度：30ms
ディスク回転数：3,600rpm
バッファ容量：256Kバイト

●MO-230S
記憶容量：230/121Mバイト
平均シーク速度：30ms
ディスク回転数：3,600rpm
バッファ容量：237Kバイト

ドライブユニットは「MO-120G」がソニー製，「MO-120S」と「MO-230S」は富士通製が採用されている。

価格は「MO-120G」が89,800円，「MO-120S」が118,000円，「MO-230S」が148,000円(どれも税別)。

〈問い合わせ先〉
㈱アイシーエム ☎06(648)4702

カラープリンタ
MJ-700V2C
セイコーエプソン

MJ-700V2C

セイコーエプソンはカラープリンタ「MJ-700V2C」を発売する。

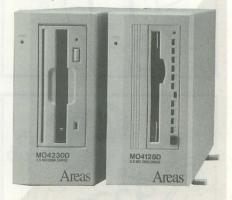
本機はスーパーファインモードで720dpi(専用コート紙使用時)の高画質印刷を実現している。通常の紙を使用する場合には360dpiの解像度となる。また，従来のグラフィック印刷に見られた走査ムラ(バンディング)をマイクロウィーブ機能により解消した。インクカートリッジは黒と3色インクの2系統になっている。ほかにも，オートフィーダを標準装備し，A4/B5各用紙で100枚，官製ハガキで30枚の連続給紙が可能。アウトラインフォントは明朝，ゴシック，毛筆体の3書体を標準装備。

大きさは，紙を出力したときの状態で470mm(幅)×525mm(奥行)×192mm(高さ)で，重さが約7.2kg。

価格は99,800円(税別)。

〈問い合わせ先〉
マッハジェットカラーインフォメーション ☎0424(99)7111

3.5インチ光磁気ディスク
MO-4128D/4230D
日本アルトス

MO-4230D/MO-4128D

日本アルトスは3.5インチの光磁気ディスク「MO-4128D」「MO-4230D」の2機種を発売する。

主な仕様は以下のとおり。

●MO-4128D
記憶容量：128Mバイト
平均アクセス速度：40ms
ディスク回転数：3,000rpm
データ転送速度：625Kバイト/s(リード時)
バッファ容量：256Kバイト

●MO-4230D
記憶容量：230/128Mバイト
平均アクセス速度：35ms
ディスク回転数：3,600rpm
データ転送速度：
920〜1,470Kバイト/s
(230Mバイト使用時)
768Kバイト/s(128Mバイト使用時)
バッファ容量：256Kバイト

価格は「MO-4128D」が122,000円，「MO

-4230Dが158,000円(税別)。
〈問い合わせ先〉
日本アルトス(株)　☎03(5820)3800

携帯情報ツール
PI-4000/FX
シャープ

PI-4000

シャープは携帯情報ツール液晶ペンコム「PI-4000」「PI-4000FX」(ザウルス)を発売した。

「PI-4000」は，従来機「PI-3000」の手書き文字の変換ペン入力に加えて，書いた文字をそのまま縮小・整理して表示するインクワープロ機能を搭載している。インクワープロ機能では1画面の行数，データ圧縮率，縮小待ち時間などが設定可能。削除や領域指定などの機能のペン入力の方法も選択できる。また，オプションのFAXモデム「CE-FM3」を利用して，同機で作成したデータを内蔵のFAX送信ソフトで送信可能。ほかにも，カレンダー，アクションリストなどのPIM(パーソナルインフォメーションマネージャ)機能が強化されている。記憶容量は544Kバイトでユーザーエリアは約355Kバイト(従来機：約157Kバイト)と2倍以上になった。

「PI-4000FX」は，「PI-4000」とFAXモデム「CE-FM3」がセットになったもの。

価格は「PI-4000」が75,000円，「PI-4000FX」が91,000円(ともに税別)。
〈問い合わせ先〉
シャープ(株)☎06(621)1221,03(5261)7271

4倍速CD-ROMドライブ
PX-43CH/45CH
プレクスター

プレクスターは4倍速CD-ROMドライブ「PX-43CH」(内蔵型)「PX-45CH」(外付け型)を発売する。

主な仕様としてはデータ転送速度が最高

PX-43CH/PX-45CH

で614Kバイト/sec，アクセス速度は220ms(ランダムアクセス時)，160ms(ランダムシーク時)，データバッファ容量が1Mバイトとなっている。インタフェイスはSCSI-2に対応。

価格は「PX-43CH」が84,800円で「PX-45CH」が96,800円(ともに税別)。
〈問い合わせ先〉
プレクスター(株)　☎03(3847)8317

ポケットFAXモデム
MC24FC5
マイクロコア

MC24FC5
マイクロコアはポケットFAXモデム「MC24FC5」を発売した。

本機はデータ通信系として通信速度は2,400bps，通信規格は国際勧告であるITU-T V.22bisに対応している。またデータ圧縮やエラー訂正機能に関してもMNPクラス5，ITU-T　V.42bisに準拠している。FAX通信系としては通信速度が9,600bps，G3規格およびClass 1/2に対応。

価格は24,800円(税別)。
〈問い合わせ先〉
マイクロコア(株)　☎03(3448)0811

無停電電源装置
BU504XIII/XLIII
オムロン

BU504XLIII
オムロンは無停電電源装置「BU504XIII」「BU504XLIII」の2機種を発売した。

両機は接続した機器の電源のON/OFFに合わせて同機の電源のON/OFFができる無停電電源装置である。出力容量は500VA(400W)でパソコン2台程度なら5分間の電源バックアップを行う。充電時間は5時間(2.5時間で80%の充電)。出力コンセントはバックアップ出力が2系統，スルー出力が1系統。バッテリーの交換時期はフロントパネルのLED表示により知らせてくれる。また，停電時や定格容量を超える過負荷時にはLED表示に加えブザーで知らせてくれる。

「BU504XIII」がボックスタイプで「BU504XLIII」は薄型タイプである。

価格は，どちらも77,800円(税別)。
〈問い合わせ先〉
オムロン(株)　☎045(411)7223,06(282)2672

INFORMATION

「世界最大の恐竜博」内イベント
DINO・PARK
ヒューマン

ヒューマンは「世界最大の恐竜博」に，体験型テーマパーク「DINO・PARK」を出展する。同展には「DINO RACER!!」「プテラノDIVE!!」などのオリジナルアミューズメントマシンが用意されている。

開催期間は1994年6月18日～9月4日，時間はAM10：00～PM8：00。場所は大阪市南港地区アジア太平洋トレードセンター(ACT)内「世界最大の恐竜博」会場。入場料は800円～2,400円(税込)で「DINO・PARK」は無料。
〈問い合わせ先〉
「世界最大の恐竜博」実行委員会事務局
☎06(201)9657，03(3543)7020

FILES Oh!X

このインデックスは，タイトル，注記――著者名，誌名，月号，ページで構成されています。夏休みももうすぐです。しっかり遊ぶためにも体調を崩さないように食生活には気をつけましょう。ひとり暮らしの人は特に注意してね。

参考文献
I/O　工学社
ASAHIパソコン
ASCII　アスキー
コンプティーク　角川書店
C Magazine　ソフトバンク
電撃王　主婦の友社
PIXEL　図形処理情報センター
マイコンBASIC Magazine　電波新聞社
My Computer Magazine　電波新聞社
LOGIN　アスキー

一般

▶みんな得するパソコン連携に挑む
異なる機種間のデータコンバータの方法をレクチャーする。パソコンLAN，データ変換ソフト，短距離パソコン通信などを利用する。――西田宗千佳・浅井聡，ASAHIパソコン，5・15号，16-27pp.

▶機械用言博物館　9
動詞をキーワードにして，パソコンし操作を解説する連載。「クリックする」をテーマにマウスの操作と考え方について述べる。――荻窪圭，ASAHIパソコン，5・15号，100-101pp.

▶最新ハードウエア情報
アップルから登場したPowerMacintoshシリーズの紹介や，緑電子の3倍速CD-ROMユニット，ソニーの28800bpsモデムなど注目のハードが登場。――編集部，コンプティーク，6月号，103-105pp.

▶趣味の基板IIX
キョーワインターナショナルの鈴木氏のインタビューをメインに，ブロック崩しについても解説する。――編集部，コンプティーク，6月号，118-119pp.

▶NEWS COLLECTORS
セガのサターンのフォルム公開，映画「ストII」の前売り開始の情報など，ゲーム業界の最新ニュース。――編集部，電撃王，6月号，22-25pp.

▶NewPRODUCTS
レーザーアクティブ用「3D MUSEUM」「3Dゴーグル」を紹介。ほかにWindowsのヴァーチャルリアリティソフトなどを取り上げる。――編集部，マイコンMagazine，6月号，42-49pp.

▶コンピュータミュージックショー・ケース
コンピュータミュージックのハードやソフトを紹介するコーナー。「Wave Blaster」「ASP」を紹介する。――編集部，マイコンMagazine，6月号，58-59pp.

▶Arcade Game Graffiti
1980年当時のアーケードゲームを振り返る。「平安京エイリアン」「サスケVSコマンダー」など懐かしい名前が登場。――編集部，マイコンBASIC Magazine，6月号，144-147pp.

▶ALL ABOUT X-DAY
余命判定装置「X-DAY」。自分の余命がどう決まるのか，アルゴリズムに迫る。――編集部，マイコンBASIC Magazine，6月号，148-149pp.

▶ゲーム考現学　1
テーマを決めて独特な雰囲気をもつ作品を紹介していく。第1回目のテーマは「蒸気と幻想」。――山田整＆"永久機関"，マイコンBASIC Magazine，6月号，148-149pp.

▶NEWS
YHPとエプソンのカラー対応インクジェットプリンタ発売，コンパックの新製品発売，980円でCD-ROM出版を実現したニュースなど。――編集部，ASAHIパソコン，6・1号，8-11pp.

▶CD-ROMパソコンの選び方教えます
CD-ROM入門者のためのガイド。パソコン各機種のCD-ROM事情の解説や，CD-ROMにからむさまざまな疑問に答える。――編集部，ASAHIパソコン，6・1号，16-27pp.

▶アジアの言葉をパソコンで使おう
日本語以外のワープロについて疑問に思ったことはないだろうか？　アラビア語，ハングル，中国語の文字構造と入力方法を紹介する。――石川京子，ASAHIパソコン，6・1号，104-109pp.

▶機械用言博物館　10
動詞をキーワードにして，パソコン操作を解説する。この回は「編集する」がテーマ。――荻窪圭，ASAHIパソコン，6・1号，112-113pp.

▶特集　パワーPC革命
PowerPCチップの実力を探る。プロセッサの概略，PowerMacintoshのレポート，また今後登場するOSの問題など，PowerPCの未来を多角的に検証。――田嶋孝行ほか，I/O，6月号，36-57pp.

▶最新日本語FEP
代表的な日本語FEP 4種を取り上げ，入力方式・変換効率・カスタマイズ・ユーティリティなどの項目について比較する。――さるさ，I/O，6月号，70-80pp.

▶特集　ディスプレイ選びのTips
高解像度ディスプレイを購入する際のポイント説明，各メーカーの主力製品の紹介など。――黒道士，I/O，6月号，92-104pp.

▶MultiMedia Watching　6
3DO発売後の評判や，マイクロソフトの販売戦略，文字多重放送の現状など。――奥野雅之，I/O，6月号，128-131pp.

▶特集I　DX4×P5×PowerPC
新世代CPUを特集する。インテルのPentium，モトローラのPowerPCなどの解説と，搭載マシンを検証する。ベンチマークテストも実施。――編集部，ASCII，6月号，241-272pp.

▶特集II　MONO・MANIA
パソコンの周辺機器特集。といってもただの周辺機器ではなく，ラベルライターやセキュリティ用品，コーヒーメーカーなどパソコンライフに登場するものを紹介する。――編集部，ASCII，6月号，281-296pp.

▶PRODUCTS SHOWCASE
エプソンのインクジェットプリンタ「MJ-700V2C」，YHPのレーザープリンタ「LaserJet4PJ」などの新発売の周辺機器を紹介する。――編集部，ASCII，6月号，308-324pp.

▶新科学対話　第6回
竹内郁雄氏の連続対談。東京大学助教授でカオス理論の研究者，金子邦彦氏を迎えて，自然界とカオスの関わりについて話を聞く。――編集部，ASCII，6月号，330-336pp.

▶INTERCOOLED　第2回
次世代ゲーム機に関するコーナー。この回は「3DO」とアタリの「Jaguar」のソフトを紹介。――編集部，ASCII，6月号，354-357pp.

▶魅惑のニューテクノロジー　第3回
最新技術を解説するコーナー。この回は，新しいバスの規格PCI(Peripheral Components Interconnect)について。――編集部，ASCII，6月号，362-367pp.

▶Digi-Ana Valley PART2　第1回
ソニーが開発中の「PS-X」について，SCEの久多良木健氏にインタビューする。――河村康文，五十嵐和久，ASCII，6月号，377-384pp.

▶なんでも相談室
「PCM」と「AD PCM」の違いについて，サンプリングの方式をいくつか解説する。――編集部，ASCII，6月号，432-435pp.

▶特集　最新ディスプレイ装置活用研究
GUI時代にマッチしたディスプレイを選ぶために，業界の最新動向と購入時のポイントをチェックする。最新機種の一覧表つき。――編集部，My Computer Magazine，6月号，13-28pp.

▶マイクロコンピュータショウ'94見て歩き
マイクロプロセッサなどの新製品を一堂に展示するマイコンショウの模様をレポートする。――丹治佐一，My Computer Magazine，6月号，36-38pp.

▶ビデオを使って身近なマルチメディア
シャープから発売されたSCSI接続のビデオデジタイザ，「CZ-6VS1」を取り上げる。付属ソフト「ライブスキャン」のレビューも行う。――高橋雄一，My Computer Magazine，6月号，47-50pp.

▶パソコン研究室
「フロッピーディスクの構造と仕組み」をテーマに，その中身と記号の読み方から各部品の材質まで徹底解剖する。――Space Club，My Computer Magazine，6月号，91-93pp.

▶フロッピーディスクの消える日？
CD-ROMの普及でフロッピーディスクはなくなるの？　記録メディアの変遷をたどりながら，次世代記録メディアの行方を探る。――山田憲一，My Computer Magazine，6月号，140-141pp.

▶ビジネスマンのための情報管理術
小型携帯情報ツール「ザウルス」とパソコンをつなぐソフト「Zaurus For Windows Ver.1」を紹介し，なにができるかを解説する。――塚田洋一，My Computer Magazine，6月号，148-151pp.

▶特集　パソコンお悩み相談室
もっとパソコンのことを知るために，ユーザーが抱きがちな35の疑問に回答する。──編集部，LOGIN，11号，131-145pp.
▶THE NEWS FILE
ソニーの次世代ゲーム機「PlayStation」の報告会のレポートや電子ディスプレイ展'94の様子など。──編集部，LOGIN，11号，146-153pp.
▶特集　リアルタイム3次元CGを可能にする次世代ゲーム機－各社の戦略と現状
次世代ゲーム機を3次元CGの視点から機種別に分析する。次世代ゲーム機6機種についての現状一覧表つき。──編集部，PIXEL，6月号，58-75pp.

X1/turbo/Z

X1シリーズ
▶キャッスル・アンド・キマイラ
3D迷路型アクションRPG。オートマッピング機能つき。──森敬雄，マイコンBASIC Magazine，6月号，114-116pp.

X68000

▶SUPER SOFT EXPRESS
新作ゲームの情報ページ。X68000用は「ぷよぷよ」「ジオグラフシール」「麻雀航海記」などが登場。新作カレンダーつき。──編集部，コンプティーク，6月号，29-49pp.
▶How to Win
X68000用はアーケードから移植された「大魔界村」を紹介。──編集部，コンプティーク，6月号，90-91pp.
▶新作予定表
各種パソコン，コンシューマ機の新作データを収録。X68000用「魔法大作戦」など。──編集部，電撃王，6月号，174p.
▶紫電光
シューティングゲームのショートプログラム。──村上剛，マイコンBASIC Magazine，6月号，117p.
▶MILKY FORCE
全方向スクロールのアクションシューティングゲーム。──高橋秀之，マイコンBASIC Magazine，6月号，118-120pp.
▶ベーマガ パソコンゲーム セレクション
「第1回全日本X68000芸術祭」の入選作品から5作品を収録した「ザ・ワールド・オブ・X68000 II」を紹介する。──板場利光，マイコンBASIC Magazine，6月号，160-161pp.
▶SUPER SOFT HOT INFORMATION
X68000用「ザ・ワールド・オブ・X68000 II」のほか，「宝魔ハンターライム10」などを取り上げる。──編集部，マイコンBASIC Magazine，6月号，別冊13p.
▶GameBox
ビングの「スーパーリアル麻雀PⅣ」を紹介。──編集部，I/O，6月号，29p.
▶AV STRASSE
マルチメディアを指向するAVマシンの最新情報。「ビデオCD for X680x0」を紹介する。──編集部，ASCII，6月号，369-372pp.
▶ONLINE SOFTWARE INDEX
大手ネットにアップロードされたソフトを紹介する。X68000用多機能電卓「calcSx.x」など。──編集部，ASCII，6月号，463p.
▶なんでもQ&A
「SX-WINDOW ver3.1」の概要と，「MUSIC SX-68K」について。──編集部，My Computer Magazine，6月号，162-163pp.
▶NEWSOFT
ゲームソフトの内容紹介。X68000用は「宝魔ハンターライム」「あすか120%」。──編集部，LOGIN，11号，10-25pp.
▶未確認クリエイターズ
ログイン大賞CG部門の入選作品発表。笹川隆幸氏の「ハニワボール」，彩人氏の「BIRDS」が1席に入選。──編集部，LOGIN，11号，172-177pp.
▶これがビデオCDだ
ビデオCDについて解説する。X68000用の「ビデオPC」の紹介あり。──編集部，LOGIN，11号，194-197pp.
▶簡単で本格的な2次元画像作画教室　5
「MATIER」を使った作画教室の最終回。スーパーリアルイラストに挑戦する。──長谷川一光，PIXEL，6月号，54，132-137pp.
▶最新X68版GNU C Compiler「XGCC」
同誌で連載中の「SX-WINDOWプログラミング」で使用するコンパイラの最新版を収録。──吉野智興ほか，C Magazine，6月号，付録ディスク，175p.
▶SX-WINDOWプログラミング　第8回
この回は構造体ビッツを用いたウィンドウの描画について解説する。──吉野智興，C Magazine，6月号，145-150pp.

ポケコン

PC-E500
▶THE競馬2
4頭の馬が繰り広げるレースの連勝の組み合わせを予想する。馬の体調表示つき。──はたらいたはらたいら，マイコンBASIC Magazine，6月号，121-122pp.

新刊書案内

どうなる?!パソコン業界
ドゥヴォラックの大予言
ドゥヴォラック著
松田　浩訳
翔泳社刊
B6判　317ページ
1,800円（税込）

「コンピュータ雑誌のコラムニスト」という職業がアメリカでは確立されている。彼らはフリーランスで，独自の情報を集め，さまざまなパソコン雑誌にユニークなコラムを書く。なかでもドゥヴォラックは多くの連載を持つ人気コラムニストである。日本では，かつてソフトバンクが出していた『パソコンマガジン』で『PC Magazine』誌の連載が読めたし，いまでは『Mac User日本語版』で『Mac User』誌のコラムが毎月訳されている。日本で定期的に邦訳が読めるコラムニストのひとりだ。
そのドゥヴォラックの著書『Dovorak Predicts：An Insider's Look at the Computer Industry』が，ビジネス新書的なちょっと恥ずかしい邦題で翻訳された（ビジネス書として売ろうとしているのだろうか）。時事的なネタが多いコラムに翻訳の遅れは致命的だが，本書は1994年2月に刊行されたものを2カ月で翻訳してしまう特急ぶり。訳や校正に一部突貫工事的なミスが見受けられるものの，このスピードは注目すべきだ。
内容はといえば，タイトルどおり，ユーザーに向けて，ドゥヴォラックがパソコン業界の今後を斬っていくもの。基本的な立場は，マイクロソフトの戦略に懸念を抱き（彼自身はOS/2とMacintoshを気に入っているらしい），特定の企業に頼ることなく，将来を見極めること。特に，現在覇権を握り，かつ，それを永続的なものにしようと画策しているマイクロソフトについては，「マイクロソフトとともに業界から消えても惜しくないもの」を引っ張り出すなど，一企業による独裁に危険を感じている。ほかのネタでも，竹を割ったようにすっぱりとパソコン業界のさまざまな事象の未来を論じていく。日米の違いはあるものの，アメリカのあとを追って導入される技術が日本にはまだ多いのも事実であり，目を通しておくのもいいだろう。予言が当たるかどうかは別にして。　（K）

吾輩はフリープログラマー
妻も子もあり猫もいる
菅　昭美著
ワードクラフト編
毎日コミュニケーションズ刊
☎03(3211)2568
A5判　207ページ
1,280円（税込）

不景気のさなか，就職が難しいらしい。そんなときの選択のひとつとしてフリープログラマーというのはいかがだろうか。
本書はあるひとりのフリープログラマーの回顧録である。というとずいぶん堅苦しいのだが，なんのことはない，私事の数々である。しかし，そのエピソードの1つひとつが面白い。プログラムを組んだ人や納期の苦しみを味わった人は，共感を覚えることは確実である。プログラムで疲れた頭を休めるにはよいかもしれない。
巻末には「あなたはフリープログラマーになれるか？」適正試験問題つき（笑）。

なんでパソコン通信やるの
大胎博善著
文藝春秋刊
☎03(3265)1211
B6判　254ページ
1,400円（税込）

本書には技術的なことはほとんど書かれていない。パソコン通信を利用して，なぜか自分の身近なものとなり，生活から切り放せなくなった人のエピソードの数々である。パソコン通信の使い方は人によって本当にさまざまである。本書から感じとれるその人々の楽しみは，パソコン通信そのものにあるのではなく，それを手段として広がっていく自分たちの世界にあるのだ。
現在パソコン通信に多くの時間を費やしている人には視野を広める意味で，そして興味のない人もパソコン通信によってどんなことができるのかを知る意味で読んでみてはどうだろうか。

QUESTION and ANSWER

よくOh!Xのカラーページで見るんですが，SX-WINDOWの背景で65536色を表示していますが，同じようにするにはどうやったらいいんでしょうか？

東京都　千葉　宏明（ほか多数）

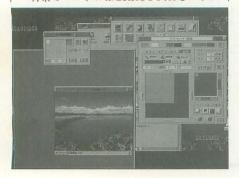
SX-WINDOWでは背景設定時に，背景に張り付けるチップの大きさが16×16ドットのときに限り，透明色がグラフィック透過色として機能します。

たとえば，パターンエディタ.Xを起動して，作成するパターンの大きさを16×16ドットに指定し，内部を透明色で塗りつぶします。次にメニューから「全選択」，さらに「コピー」を選択してみてください。

そしてコントロールパネルを開き，背景設定を選びます。パターンをペーストしてみるとグラフィックが透けて見えるのがわかると思います（あらかじめ，なにかグラフィックを表示しておいてください)。これを使用すれば65536色の背景を敷くことができるわけです。

ただし，理屈で考えてもわかると思いますが，この状態でグラフィック画面を扱うアプリケーションを起動すると背景は壊れてしまいます。65536色のグラフィック画面は512×512ドットの大きさしかありませんから，ウィンドウ以外のところにゴミが出てしまうのもしかたないことです。

SX-WINDOW使用時にはグラフィック画面の使用頻度はそう高くありません。一般的なアプリケーションでグラフィック画面を扱うものというのはEasypaintとキャンバス，CGAウィンドウ，ライブスキャンくらいのものですから，使えなくても実害はないといえるかもしれません。こう，割り切れば背景専用と考えても不都合はないでしょう。

背景のロードにはGRROOT.Xというフリーソフトウェアなどを使用していますがなければ普通のキャンバス.Xでもかまいません。確実にG-RAMにデータを読み込めればなんでもかまいません。GL3ファイルのロードくらいならSX-BASICでも簡単に記述できるでしょう（たぶん使う人はいないでしょうが)。

もっと一般的な手法としては，キャンバス.Xを使って画像をロードしたあと，メニューの「実寸大モード」で画面いっぱいに画像を表示して，キャンバス.X, GRW.Xをそのまま終了してください。

起動時に自動的に背景を設定したいという場合は，GRWウィンドウの背景としてなんらかの絵を登録しておくという手があります。そのように設定したうえで，SX-BASICで，

```
fock("GRW.X")
```

のようなプログラムを書いて自動起動指定しておけばよいでしょう。SX-BASICのプログラムを自動実行指定するには，SX-BASICのウィンドウ内で，

```
autoexec
```

というコマンドを実行したあとで編集中のファイルをセーブします。GRWウィンドウを起動するだけなら，なにもしなくてもSX-WINDOWが起動してくれますが，画像を完全に表示するには必ず一度アクティブな状態になる必要があるのでこのようにしているわけです。

SX-WINDOWが立ち上がったらGRWは手作業で終了してください。本来ならSX-BASICだけですべて制御できるはずだったのですが，子プロセスで立ち上げたキャンバスがGRWウィンドウ外に開いてしまったり，SX-BASICのマニュアルにあるTSEnd()は実は実装されていなかったりということでうまくいきませんでした。

ちなみにGRROOT.Xとは，GRWウィンドウを使用せず，直接G-RAMにグラフィックデータを読み込むためのツールです。ひなまつりPRO-68Kの隠しファイルにこれを使ったSX-BASICのサンプルプログラムが入っていたのですがお気づきでしょうか。それはGRROOT.Xを使って背景を10分ごとに切り換えるというものでした。メモリの状態がゴチャゴチャしてくるとコマンドラインの受け渡しに失敗しますが，まあそこそこには使えます。プログラムは不可視属性になっているだけなのでGRROOT.Xをお持ちで興味のある人はATTRIB.Xで不可視解除して使ってみてください。

そのほか，編集室で使用しているSX-WINDOWの背景をディスクに入れてほしいという要望も多数ありましたが，だいたい平均して300Kバイト以上のPICファイルになっているので付録ディスクへの収録などは，まず不可能でしょう。世界的に有名な某ネットワークにフリーデータとして流れていたものがもとになっていますが，著作権や肖像権とか怪しい部分もありますし。

なお，来月号ではこのような65536色背景を使ったプログラムやSX-WINDOW用のPICローダが掲載される予定ですのでご期待ください。

6月号の「メガディスプレイへの道」はたいへん面白かったのですが，うちのディスプレイは初期型（CZ-600D）で6年も使い込んだらすっかり暗くなってしまいました（新しいのを買えというお告げかな？)。ところで本体改造での高改造はX68030ユーザーだと，なにも69MHzを抜かなくても50MHzを100MHzにしてもいいんですよね？

神奈川県　井手　裕二

X68000CompactXVI以降の機種ではVGAディスプレイ（というよりは液晶ディスプレイ）対応モードとしてVGAモードが用意されています。そしてこのモードを作り出すために，ほかのX68000よりは1個余分に50MHzのオシレータがついていたのです。

実は，メガディスプレイの企画も当初の予定ではX68000CompactXVI以降を対象として50MHzのオシレータを取り替えようというものでした。VGAモードを使うような酔狂な人は多くないだろうということで，Vキーを押して立ち上げればSX-WINDOW専用高解像度モードで起動する……となるはずでした。

しかし，X68030に使用されているオシレータでは1番ピンでオシレータ自体の動作/非動作を切り換えているらしく（通常はNCまたは分周クロック出力など)，それに対応したオシレータでなければうまく動作切り換えが行われずに画面が流れてしまいます。おそらく複数のクロックがCRTCに入っているものと思われます。

1番ピンで動作を切り換えるようなオシレータがみつかるか，そもそも100MHzのものがあるのかという心配から始めなければなりません。ですから試しにやってみて動けばよしとなりますが，そうでない場合は2つのオシレータを取り外さなければなりません。

また，オシレータはその性格上，安易に切り換えスイッチなどをつけると特性が悪くなるので，改造する場合にはある程度割り切った使い方をすることが必要になります。

通常のディスプレイで69.5MHzを72MHzに換えただけでもわずかにちらつきを減らすことができますが，その程度の効果で高いリスクを抱えるべきではないでしょう。

もともとOSCIAN2などはかなり熱を持つLSIなのですが，100MHzを積んだ場合などは時間がたつと画面の合成部などにノイズが乗ることがあるようです。過熱するようなら，ヒートシンクなどを取り付けてください。また，ディスプレイにもよりますが，特定の値をセットするとCRTCをリセットするまでは画面になにも映らなくなってしまうこともあります。

こういったCRTC高速化改造を行うと画面が広くなるだけでなく若干の副作用もでてきます。たとえば，CPUが16MHz以上の場合，垂直同期期間が短くなることやG-RAMが軽くなる関係でゲームなどが異常に速くなってしまうことがあります（フロントポーチやバックポーチが異様に長くなるのでG-RAMのウエイトがかなり減るのだそうだ)。ノーマルな動作環境を残したい場合にはこのような改造はしないほうがいいでしょう。

無改造，標準ディスプレイのままでもこれまでよりもはるかに使いやすいモードも開発されていますので，普通の人はあまり無理をする必要はないでしょう。

高解像度はともかく，SX-WINDOWのスクロール座標関係を補正するツールがなければこういったモードもまともに使うことはできません。そういったものはおそらくSX-WINDOW ver.3.1のパッケージでサポートされるのだろうと思われますが，SX-WINDOW ver.3.1発表後にはOh!Xでも独自のものを提供する予定です。

(U)

“マッドストーカー”についての質問です。このゲームHDにインストールしてプレイできますが，HDから立ち上げると，PCM8が使用できなくなってしまいます。FDから立ち上げれば使用できるのですが，なぜなのでしょう。このゲームをHDから使う場合，"AUTO_HD.BAT"とコマンドから入力しますが，このファイルに変更を加えなければいけないのでしょうか？ しかも，このファイルは不可視になっているので覗けません。どのようにすればHDからでもPCM8が使えるのか詳しく教えてください（できれば不可視ファイルの覗き方もお願いします)。

東京都 細田 修宏

結果からお話ししましょう。マッドストーカーをインストールしたディレクトリに以下のリスト打ち込んで，それを実行すれば，マシンがなんだろうと，有無をいわさずPCM8を組み込んでちゃんと起動します。

```
sys\assdrv /p
path a:\;a:\sys;
pcm8 -I4
zmsc-M -W0 -t0 -b\pcm_
dat\PCM1.zpdm_title > NUL
main
```

不可視属性のついたファイルというのは，文字どおり，普通では見えなく，扱えないファイルです。dirでも見れませんし，copyもtypeもできません。しかし，これはdirが見せなく，copyがcopyできなく，typeが内容を見せなくしているわけなので，dump.xなら中を覗くこともできますし，copyall.xならcopyすることもできます。

実はauto_hd.batというファイルは，不可視属性どころか，リードオンリー（書き込み不可）属性もついています。このようなファイルから属性をはずし，普通のファイルにするためには，標準コマンドのattrib.xを利用します。

auto_hd.batを通常のファイルにしたい場合，コマンドラインから，

```
attrib -H -R auto_hd.bat
```

と打ち込むことで，このファイルから属性がはずされ，普通のファイルになります。-Hというオプションは不可視属性の解除で，逆に+Hという属性をつければ不可視属性をつけることが，すなわち不可視のファイルにすることができます。また-Rというオプションはリードオンリー属性をはずす指定で，逆に+Rとすれば読むことしかできないファイルを作ることができます。

さて，これらの属性というのは，簡単に標準コマンドで解けてしまう基本的で簡単なファイルプロテクトです。簡単に解除できるようなら，プロテクトの意味がないように思えますが，簡単に解除でき付加できるという意味で便利な点があります。

まず，リードオンリー属性は，間違えて消してしまいたくないファイルにつけるのが常識です。アプリケーションやツールなどは，過って消してしまっても再インストールすればよいですが，一生懸命作ったconfig.sysやautoexec.batなどは過って消されてしまっては困りますよね？

このようなファイルに，リードオンリー属性がついていれば，もしも過って，del *.*などと，タイプしても消えることがありません。リードオンリー属性はOS的に保護されているので安心できますよね。

一方，不可視（ヒドゥン）属性はファイルがいろいろあるとき，見えないほうが都合がよいものに利用します。見えなくても，エディタなどで読むことができますし，また，書き替えることも可能です。ですから，定義ファイルなどをすべて不可視にしておくと，dirを取ったときにすっきりと美しくまとまるので，見栄えがよくなるというわけです。

(瀧 康史)

質問にお答えします

日ごろ疑問に思っていること，どんなことでも結構です。どんどんお便りください。難問，奇問，編集室が総力を挙げてお答えいたします。ただし，お寄せいただいているものの中には，マニュアルを読めばすぐに解答が得られるようなものも多々あります。最低限，マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名，システム構成，必要なら図も入れてこと細かに書いてください。また，返信用切手同封の質問をよく受けますが，原則として，質問には本誌上でお答えすることになっていますのでご了承ください。なお，質問の内容について，直接問い合わせることもありますので電話番号も明記してください。

宛先：〒103 東京都中央区日本橋浜町3-42-3
ソフトバンク株式会社出版部
Oh!X編集部「Oh!X質問箱」係

FROM READERS TO THE EDITOR

眩しい陽射しのなか，公園を歩いていく。目に映る緑の葉が茂る樹木は，夏の匂いを感じさせる。ビルの窓から見える河の水面にも陽光が反射する。梅雨のときに外で遊べなかった鬱憤をいまこそ晴らすのだ。いざ，街へ，山へ，海へ！

◆「言わせてくれなくちゃだワ」ももう9回目ですか。早いものですね。Oh!Xを読み始めたのが1985年6月号，S-OSのスタートした号でした。「ちゃだワ」や「GAME OF THE YEAR」が始まったのも，そのちょっとあとのことでした。ずいぶん月日は流れましたが，それらの企画は続けられ，有田さんの連載や「猫とコンピュータ」も健在です。変わっているようで変わっていない不思議な雑誌ですね。　福居　毅至(22)兵庫県

これからもOh!Xをよろしくお願いします。

◆5月号の「DōGA CGアニメーション」の人の顔は，はっきりいって怖い。なんか生きているみたいでゾーッとしてしまった。バスやロボットの行進のほうが安心する。

鈴木　勝治(37)愛知県

生きている人のデータを取り込んでいるんだから生きているみたいで当然なんですが，ということはマンデル北尾さんの顔が怖いってことでしょうか。

◆「PUSH BON!」面白いですね。あのキャラクタの動きや表情がかわいいと，うちの奥さんも気に入って，はまってしまいました。あのキャラクタには名前はないのでしょうか。うちの奥さんは「ラビー」やら「関根くん」(わかります？)などと呼ぼうとしています。ただ残念なのはジョイスティックでゲームするときにGIVE UPできないことですね。　横堀　正敏(30)埼玉県

たくさんの皆さんにご好評をいただきありがとうございました。あと，質問の多い36面ですが，解けません。どうもすみませんでした。

◆おおっ！　5月号にはアンケートハガキが2枚。これでプレゼントの当たる確率は2倍だと思ったが，みんなで送ったら変わらないね。

伊南　尚幸(18)山形県

そんなことはないですよ。ほかにも2枚ついていたと書いてきた人はいたけれど，当たる確率は1.5倍にはなったでしょう。

◆なにか付録ディスクのタイトルを見ていると「日本全国酒のみ音頭」を思い出す。ということで11月号に期待。　荻野　潤(19)埼玉県

さて，11月号にディスクがつくとしたら，そのタイトルは……。知りたい人はカラオケにでも行ってみましょう。ただ11月号にディスクがつくかどうかは保証の限りではありませんが。

◆5月号78ページの石田殿。ファイル名は短いに限る(きっぱり)。以上。

井川　大介(25)東京都

人それぞれ好みがあるってことですね。個人的にはファイル名を見ただけで内容が判断できるとありがたいですけど。

◆5月号67ページの松田さんへ。個人でFDDユニットを買うよりメーカーでFDDユニットを交換するほうが安いのは，シャープがFDDユニット自体を生産していないことをはじめ，いろいろ事情があります。別にパソコンに限ったことでもシャープに限ったことでもありません。私からは詳しい説明は省略しますが，松田さんが電話なさったサービスステーションに聞いてくだされば納得いく説明があると思います。[電気製品修理業者より]　鵜沢　則之(36)千葉県

そうだったんですか。よく知らないこととはいえ失礼しました。

◆5月号の満開製作所の広告には，X68000 CompactXVIの24MHz化は4/31必着とか書いてあるけど4/31は存在しませんね。まるで細川元首相のコメントのようです。

山中　政宣(20)三重県

いや，池袋の某空間では実際に4/31が存在したとか(ウソ)。

◆表紙をめくって「新機種だ！」と思ったらMacintoshだった。　鈴木　勇(19)北海道

ほかにも多くの方が「CZ-6VS1」の広告にあるMacintoshを新機種と勘違いしてしまったようです。

◆「ちゃだワ」に載ったのがとても嬉しかった。次の日，学校にもっていくと自分の書いた内容を見られて，みんなにバカにされた。

綾　雅樹(17)広島県

そんなことでくじけないで，来年もアンケートを出してくださいね。

◆ベッドの下の斧男は怖いですね。私も小学4年生の頃，デパートのトイレでなんと「個室直立ナイフ所持ヨダレたらし男」を見ました。友人が1人逃げ遅れて捕まりましたが，警察を呼んで危機一髪でした。トイレにはなにが潜んでいるかわかりません。気をつけましょう。

山田　公則(18)神奈川県

世の中あらゆるところに危険が迫っているみたいです。ほら，いま本を読んでいるあなたの後ろにも……。

◆本当はプログラマの私ですが，最近はプログラミングから違う方向へ流れていってたりします。気がつけば，ほかのグループに回され，キーパンチャーと化している。この不景気のなか，夜中の2時，3時まで仕事するほど……というのは，ありがたいといえばありがたいのですが……眠い。　山口　桂司(21)茨城県

どんなご時世であれ，なかなか自分の時間が取れないのは困りものですよね。ふう。

◆「三國志IV」(PC-9801版)で中国統一をしました。IIあたりから他国との同盟が重要な戦略のひとつですが，決して同盟を結びません。他人(特にコンピュータ)なんて信用できないし，自分の行動を制限することにもなります。こんな考え方だから私は友人が少ないんだな……。

◀岡村　直也　兵庫県
第4話。このシリーズのテーマはラブコメだそうです。学の最後のひと言がとってもいいのですが，実際にいうとしたらけっこう恥ずかしいですよね。

◀日高　光代　宮崎県
こんな夢を見るなんてなにか過去で苦労したことがあるんじゃないですか。たとえば，テニス部だったときに重いコンダラ引かされたとか(古い)。

中村　健(24)埼玉県

でも中村さんが選んだ人なら，きっと友達を裏切らない，いい人でしょうね。

◆シャープもそのうちにX680x0を通販するのでは……。　岡村　昌彦(20)静岡県

でも，それが新機種だったら通販も悪くないかもしれません。

◆「ジオグラフシール」に魅せられて発売以来1カ月以上，毎日出かける前にタイムアタック＆ハイスコアをたたきだすのが日課になっています。記録は17分43秒で125万点。なかなか熱いです。次の「GAME OF THE YEAR」の大賞はこれで決まり……かな？　フフフ。ちょっと早いし，コナミの出方次第か(安易な考え)？　そろそろエグザクトさんに大賞をあげたい今日この頃。

鈴木　政宏(21)宮城県

あまり熱くなりすぎて学校に遅刻しないように気をつけてくださいね。

◆付録ディスクが手に入った。電脳倶楽部もちゃんと届いた。TAKERUのクーポンが当たった(1万円分)。皐月賞も当てた(もちろん馬連)。「スーパーリアル麻雀PⅣ」の資金ができた。でも愛機は修理中(悲)。　井村　英二(22)滋賀県

早くパソコンが戻ってくるといいですね。さもないと愛機に投資をする前にほかのことでお金がなくなったりして……。

◆これはロマンスカーの中で書いている。なにせ大学は神奈川県，家は春日部。通学時間は3時間。はぁー，きつい。それはさておき，私はまだX68000を買っていない。早く触りたいぞー。母よ，早くX68030を注文してきなさい(母はシャープに勤めている)。半田　将義(20)埼玉県

ロマンスカーって，確か全席指定だったような気がします。ということは学校には座っていけるんですね。うらやましい。

◆X680x0を持っていないにもかかわらず，マンデル君にひかれて買ってしまいました。

中村　隆喜(21)大阪府

ひょっとして危険な人なのでは！

◆春の山菜シーズン！　私は毎年この時期になると，有給取って山菜採りにいくんです。その日は朝から晩まで野山を駆け巡り，山菜を採ります。地元の人は，こんな私を「栄養失調のターザン」と呼んでいます。

大久保　明弘(21)岩手県

山菜採りをした翌日は「筋肉痛のターザン」となっているのでしょうか。

◆駅前でなにか配っていたのでティッシュかなと思って受け取ったら耳かきだった。なぜ耳かきを……。　志摩　憲(20)大阪府

え，だって耳かきをすると幸せになりませんか(ちょっと本気)。

◆嫁さんをもらった。なぜか家事が倍に増えてしまった。うちのX68000SUPERはホコリをかぶっている。……涙。　古野　泰(25)大阪府

X68000との時間が取り戻せることを心よりお祈りしております。

◆ある雨の日，農学部のキャンパスを歩いていると，いつも放し飼いにしてあるアヒルが水た

▲黛　拓也　神奈川県
細かいことですが，あすかが投げている試験管は確か3本だったと思います。しかし3本も試験管を持ってあんな劇薬をよく投げられますよね。

▲平　智征　神奈川県
「ぷよぷよ」の面白さに引かれて久々にイラストを出したそうです。アルルがとっても可愛いですね。それと右の黒光りしてる物体はやっぱりぷよ？

まりで水浴びをしていた。いい気なもんだと思いながら水たまりの上を，ぴょんと飛ぶとふくらはぎに痛みが走った。何だ!?と思って見るとアヒルが足に攻撃を加えていた。アヒルの名誉(？)のためにとっとと逃げ出したが，「アヒル，人間を襲う」とタイトルだけは大事件でした(笑)。でもかわいかったな。

手嶋　和徹(22)鹿児島県

逆でなくてよかったですね。というのは，人間が生きてるアヒルの足にかみついたらタイトルは「人間，アヒルを襲う」と一見おとなしいですが，実際の光景はけっこう怖いと思いませんか。

◆とうとう私にも就職活動の時期がやってきてしまいました。自分がいったいなにをしたいのか，なにに向いているのか，考えれば考えるほどわからなくなります。いまのままが続けばいいのですが，世の中そんなに甘くもない。就職したらいままでのようにはX68000をいじってられないんだろうな。就職先としてOh!X編集部はいかがでしょうか？　堺　和幸(21)宮城県

私どもに決定権はありませんが，とりあえずソフトバンクに入社してみてはいかがでしょうか。ただ，もしかしたら競争率が高くて配属は難しいかもしれませんが(？)。

◆4月17日，朝日新聞に我が大学においてのアルコールパッチテスト(酒が飲める体質か調べる)が載った。私はその係員をやっていたが，マスコミ関係らしき人が何人か来て写真を撮っていった。隣で同じく係員をしていたやつは「これで俺も芸能界デビューだぜ」とか，わけのわからないことをいっていたが，記事だけで写真は載らなかった(俺はなんも期待してないぞ)。

須佐　英之(19)千葉県

写真を撮るところには意味なくX68000を置いておきましたか。マスコミへの露出を高めるためにいつもX68000を携帯していましょうね。そのためにキャリングハンドルがついているんですから(ウソ)。

◆最近「LEE」のCMに感動している。たった15秒間で自分が強くなったような気がするのが不思議だ。　田添　政勝(24)兵庫県

そのあとでなにかを殴って，手にケガなどしないように気をつけてくださいね。あくまでも強くなった気がするだけですから。

◆中日×巨人戦(名古屋。ココがポイント)を友人と見ていたときのこと。友人「コレ誰や？」。私「コトーとかいう外人」。背中の「COTTO」の字を見て，友人「Nつけたらコットンや！」。そのあとホームランを打った。さすがナガシマ，「WILLOW」でつったな。亀田　徳隆(18)香川県

じゃあ，今年は巨人が優勝して，報酬に満足できないコトーは来季は別の球団にいってしまうのでしょうか。

◆美少女戦士から赤ずきんチャチャに転んでしまいました。変身シーンまでは面白いので，見たことのない人も一度見てみませう。これのおかげで，また当分の間は社会復帰できそうにありません。　津村　忠蔵(19)佐賀県

このままだと社会復帰したくてもできなくなるかもしれませんね。噂では1999年7月には……。

◆とーとー会社を「クビ」になってしまいました。「そろそろ他人事ちゃうよなァ」と思っていたら……。次の職は果たしてあるのか？　心境は「誰か雇ってくれ〜！」って感じです。それはそうと「ジオグラフシール」のステージ4でシークレットボーナス50万点もらったことがあるんですけど，どうやったらもらえるのかわかんないです。　中矢　史朗(23)愛媛県

ただ働きでよければOh!Xに仕事はありますが，いかがですか。

◆専門学校を中退した。どうして来年度は授業数が半分になって授業料が40万円も上がんねん。来年はサクラサカセテミセルゼ……(涙)。

杉浦　晃規(19)愛知県

専門学校の経営も苦しくなってきているんでしょうか。

◆食べ物の好き嫌いをなくそうと思い立ち，とりあえずトマトにチャレンジ。……でも……やっぱ……まずい。「美味しんぼ」に出てくるようなデリシャス・トマトを食べてみたいな〜と思う今日この頃。　森山　茂雄(19)熊本県

トマトってそんなにまずいっていうほど味がしますか？　でもデリシャス・トマトは食べてみたいですね。食べた瞬間は「豊潤

て瑞々しくて……」なんてやっぱりいいたくなるんでしょうか。

◆ひとり暮らしも慣れましたが，さっそく飢えております。最近では学食でも高いと感じるようになってしまいました。ここ2，3日の食費は1日当たり2，300円です(2千3百円じゃないぞ)。　中澤　巧爾(18)京都府

倒れないように気をつけてくださいね。どうしてもだめなときは外で倒れることをお勧めします。部屋の中だと誰も気がつかないってこともありますから。

◆デパートのUFOキャッチャーにてぷよぷよスリッパなるものを見つけ，なにげなくプレイ。……取れた！　上機嫌で帰宅。だが，スリッパは片方では意味をなさないことに気づく(笑)。数日後，同じタイプのスリッパを求め再びプレイ。500円使って取る。上機嫌で帰宅。……だが，このとき鉄之助はスリッパのサイズが小さいことに気づいていなかった。

平野　鉄之助(18)長野県

スリッパが小さいことに気がついた鉄之助は，その後スリッパの似合う可愛い女の子を捜し歩いたとか(冗談)。

◆うふふ，5月号は18日に手に入りましたよ。なにしろ，4月から福岡に住んでいますからね。連休に山口へ帰ったら愛車のアルトはエンジンがかからなくなっているかもしれません。今年で10歳ですから。　藤田　敬三(19)福岡県

すでに連休は過ぎていますが，アルトはすねたりしていませんでしたか？　早く福岡にもっていけるといいですね。それとも裏切って新車でしょうか。

◆5月号の欄外の話は，前のページのほうは北に住んでいる人で，後ろにいくほど南のほうに住んでいる人なんですね。いま気づきました。

曽川　秀和(19)東京都

残念ながらちょっと違います。後ろのページを見ていくと順番にずれがあります。本当はきっちり順番にしたかったのに……。

◆社会人になって，いま一番いやなことは帰りの満員電車だ。困ったな。

関　喜視(20)神奈川県

では，満員電車を避けるために少し遊んでから帰りましょうか。場所？　それは自分で探してくださいね。

◆もうすぐSC-88が来るのでウハウハです。死んでなければまた来月(笑)。と思ったら友人が運転する車に乗っているとき，体感リッジレーサーを味わいました。車は横滑りして溝にはまりましたが，だれもケガはありませんでした。楽しかったです。　河野　裕文(19)富山県

誰にもケガがなかったから「楽しい」ですみますが，壁などへの激突はゲームのなかだけにしたいものですね。

◆編集の皆様もそうだったかもしれませんが，ゴールデンウィークは本当に1日も休めませんでした。しかも毎日定時よりちょっと遅れて出勤，帰宅は午前サマという状態……きつかったです。まあ私はいいんですけど入社1カ月にも満たない新人さん4人につき合わせてしまいました。ごめんねーみんな。高橋　明(23)東京都

ごめんなさい。あやまることもないんですが，編集部は少し休ませていただきました。

◆よく5月病と聞くけど，早い話がぽかぽかしているから眠たいということなのだ。

服部　直幸(21)広島県

そうか，ということは私の場合1年中5月病かもしれません。

◆某CMに出ているアルシンドの日本語が妙に聞き取りやすくなった。彼がうまくなったのか，はたまた私たちの耳が学習したのか……。

喜多　清高(24)兵庫県

いわれてみれば確かにそんな気がします。きっとどっちも正しいと思いますが，アルシンドは意味がわかってしゃべっているのでしょうか。

◆家には猿が1匹いて，これがくだらないことを覚えてきて困っています。朝早くから私を叩き起こしておきながら，自分はそのあとにすぐに寝てしまいます。いいかげんにしろよな！

金原　真也(26)新潟県

猿に怒っても仕方がないので放っておきましょう。でもその猿ってひょっとしたら言葉をしゃべったり，マニキュアを塗っていたりしませんか。

◆単車に乗って40km/hで走っているときに左折車に巻き込まれてしまいました。左足がとても痛くて，救急車で病院へ運ばれました。これで家族全員(5人)が救急車に乗ったことになりました。皆さんも気をつけてください。

風呂本　真(25)広島県

家族みんなでお祓いでもしてもらったほうがいいんじゃないでしょうか。

◆うちの学校に柴田淳という先生がいる。も，もしや……？　埣本　義明(15)茨城県

そんなことはないはずですが，ひょっとしたら……。

◆私の通う埼玉大学では「教養学部」は留年がないので楽勝です。けっこう珍しいでしょ。

川島　利之(19)埼玉県

ツケというのはあとからやってくるものです。いまのうちに無理のない単位修得をお勧めしますよ(経験的に)。

◆Oh!Xの付録ディスクは名前がついていていいですね。名前でディスクの内容を思い出すことができます。でも5月は「黄金週間」「こいのぼり」ときましたが，あとが続くのでしょうか。

倉持　竜太(20)神奈川県

ご心配いただきありがとうございます。実は5月号を心配する前に10月号あたりが問題になりそうです。なにかいい名前がありましたらよろしくお願いしますね。

◆ふと「スパII」の対戦がやりたくなったので近所のゲーセンに行き，適当に対戦台から乱入し見知らぬ何者かをベガでぶちのめした。勝ち誇った私は相手がどんな顔をしているのかとチラリと覗いてみた。すると相手は女性だった。どうしよう……。　城石　政信(18)兵庫県

1．黙って逃げるように立ち去る。
2．無視して続ける。
3．100円を彼女の前に置いていく。
4．お友達になる。

さて，あなたはどれを選びますか。

◆今年もやっとF1が始まりました。でも，また月曜の学校がきつくなる。

神保　公一(18)東京都

好きなものを見ていての夜更かしだったら平気ですよね。

◆絵の勉強をしに学校へ行こうと思い，沖縄から大阪へ引っ越しました。こちらは桜がキレイでいいですねえ(いつの話だ，いつの)。沖縄の桜はなんかミョ～にタフで，あんなに見事に花びらが散りませんよ。　城間　裕樹(20)大阪府

沖縄の桜ってそんなに違うんですか。なんか考えれば考えるほど，どんなものか想像がつかないんですけど。

◆我が大学が追再試受験料を値上げしたり，プリント，大学の教材を減らすなどしている。いきなり学長も変わったし……。もしかしたら数年後大学が消える(つぶれる)のではないかと本気で心配している。　角谷　光憲(20)愛知県

そう簡単にはつぶれないと思いますが，母校にはやっぱり残っていてほしいですよね。

◀越智　文昭　愛媛県
こんな人形が枕元に置いてあったら怖くて眠れそうにありませんね。眠ってしまったらなんか悪い夢を見てしまいそうです。私は見てしまいました。

◀岩本　雅　愛知県
そういえばX68000では最近RPGって発売されていなくて，ちょっと寂しいですね。左のモンスターの上のほうの目は隠さなくていいんですか。

神谷　正樹(20)愛知県

寝てるあいだに口の中にでも入ってきたら……。とても寝てられませんね。

◆25歳になって……仕事にもゆとりができた。熟考した末，X68000を購入した。自分の選択が間違っていなかったことを確信した。その一方で……「セーラームーン」にはまった(25歳にもなって，なぜ？)。　石田　正浩(25)北海道

はまるときなんてそんなもんです。理屈ではありませんからね。

◆本棚を整理していたら，Oh!Xのバックナンバーが3年分落ちてきた。すごい音がした。ほとんどが頭に当たってから床に落ちた。痛かった。

長谷川　常吉(33)三重県

3年分でよかったですね。これが創刊号から全部落ちてきていたら痛いだけではすまなかったかもしれません。

◆東京に引っ越すことになり，ベッドのカタログを見ていた。すると“パソコンラック付ベッド”の写真に載っていたのはX68000用モニタとX68000PROのキーボードであった。また，同じカタログの中で，本が数冊置けるパソコンラックのところにあったのはHuman68kのマニュアルであった。このカタログを作った会社にはX68000ユーザーがいるに違いない。

植木　正幸(25)神奈川県

写真の小道具になるとはさすがはX68000といったところですね。でも本体が載っていないのは少し寂しいですね。

◆昨年の12月に結婚しました。結婚する前からわかってはいたのですが，主人がすっごくパソコン好き。暇さえあればいつもパソコンの前に座っているのです。私が話をしても「ウン，ウン」と返事をするだけで，話なんて聞いていません。パソコンに夢中になる主人を見ていて，パソコンにやきもちを焼いてしまい，「パソコンと私，どっちが好き？」とけんかもしました。でも，これではいけないと思い，いままで毛嫌いをしていましたが，パソコンと仲よくなろうとやり始めました。いまでは，2人で仲よくパソコンの前に座っています。Oh!Xも読み始めました。パソコンも少しずつ使えるようになり，大変活用しています。もうパソコンにやきもちなんて焼きません。　匹野　みか子(22)大阪府

いろいろと大変なこともあるかと思いますが，夫婦で仲よくパソコンライフを楽しんでくださいね。

◆なぜかは知らないが，ハガキ＆定期購読の振替用紙が2枚。これはもしかして1人2枚ずつアンケートを書かせて一挙に読者が倍になったように思わせるための陰謀か？　はっ！　じゃあ振替用紙が2枚っていうのも俺様に2冊分定期購読しろとのことなのか!?　でもすでに生協で定期購読しているから3冊分か。ちょっとヤダなあ。　森口　賢(21)福岡県

あ，ばれてしまいました。今度から2冊でセットになってお安くなりますのでよろしくお願いしますね(ウソ)。

◆4月某日，卒研のテーマに「確率現象の考察」を選ぶ。4月末日，BASICでプログラムを作り教授に見せる。教授曰く「これで論文が書けるなあ」。本当にそんなプログラムでいいんですか？

村橋　裕幸(20)岐阜県

自分で納得のいく卒業研究になるようにがんばってくださいね。

◀鈴木　貴久　神奈川県
タイトルにあることは鈴木さんの自宅の話なんですね。来年のいま頃は本当の意味での新しいXファミリーが増えているといいですね。

ぼくらの掲示板

●掲載ご希望の方は，官製ハガキに項目(売る・買う・氏名・年齢・連絡方法……)を明記してお申し込みください。
●ソフトの売買，交換については，いっさい掲載できません。
●取り引きについては当編集部では責任を負いかねます。
●応募者多数の場合，掲載できない場合もあります。
●紹介を希望されるサークルは必ず会誌の見本を送ってください。

仲間

★MZ-2500/2800を対象としたディスクマガジンを発行しているサークル「星くずばこ」です。SuperMZのことならゲーム，音楽，グラフィック，ツール，何でもござれ。昨年会員制に移行し，多くの熱きユーザーが参加しました。さらにたくさんのMZユーザーに楽しんでいただくべく，会員募集です。90円切手6枚を同封して下記までご連絡ください。折り返し入会案内と最新号のディスクをお送りいたします。〒807　福岡県北九州市八幡西区星和町24-54　山ノ内方　星くずばこ

★サークル「ICP」では現在開発しているRPGのグラフィック関係のスタッフを大募集しています。興味をもたれた方(グラフィック以外の人でもOK)は下記の住所までご連絡ください。その際には住所，氏名，年齢と自己PRを書いてください。〒565　大阪府吹田市山田西2-4-A1-408　米村　貴裕(19)

売ります

★X68000ACE/PRO用の1Mバイト増設RAMボード(内蔵用)「CZ-6BE-1B」を7,000円(送料込み)で売ります。説明書，付属品あり，箱，保証書はありません。連絡は往復ハガキでお願いします。〒343　埼玉県越谷市南越谷1-6-62 コーポ南越谷D-310　川島　康平(19)

★システムサコムのMIDIボード「SX-68MII」，ローランドの音源モジュール「CM-64」，ローランドのサウンドライブラリよりギターカード「SN-U110-07」，ドラムカード「SN-U110-10」，以上すべて箱，説明書，付属品つきで70,000～75,000円で売ります。連絡は往復ハガキでお願いします。千葉周辺の方なら直接届けます。〒262　千葉県千葉市花見川区さつきが丘2-10-4-403　玉尾　賢吾(21)

★ディスプレイ「MZ-1D22」を5,000円(送料込み)で売ります。ただし，本体のみで前ブタが壊れています。連絡は往復ハガキでお願いします。〒467　愛知県名古屋市瑞穂区師長町1-2-506　山内　紀章(22)

★HAL研究所のハンディスキャナ「HGS-68」を15,000円(送料込み)で売ります。説明書，付属品あり，箱なしです。連絡は往復ハガキでお願いします。〒890　鹿児島県鹿児島市唐湊4-13-16　児玉　秀樹(28)

買います

★X68000XVI専用増設RAMボード「CZ-6BE2A」を25,000円で，増設RAM「CZ-6BE2B」を20,000円で買います。完動品であれば，箱や説明書がなくてもかまいません。連絡は官製ハガキでお願いします。〒237　神奈川県横須賀市追浜本町2-50-242　河路　弘司(23)

バックナンバー

★Oh!Xの1990年3，10月号を各1,000円で買います。連絡は官製ハガキでお願いします。〒570　大阪府守口市金田町5-15-18　宮本　武雄(29)

編集室から from E・D・I・T・O・R

DRIVE ON

このコーナーでは，本誌年間モニタの方々のご意見を紹介しています。今月は5月号の内容に関するレポートです。

●そういえば，横置きPROシリーズの新機種ってちっとも出ませんね。ASCIIの最新主要機種一覧にはちゃんとX68000 PROIIの名前が載っているのでPRO系の生産ラインは残っているようです（?）。X68000 XVI PROとかX68030 PROとか発売されてもおかしくはないのですが……。もっともPROシリーズの最大の利点である内蔵4スロットもSCSI標準装備，メモリ，コプロ内部増設のX68030の前にはあまり意味はありません（MIDIはRS-MIDIでなんとかなるし）。そこでポイントとなるのは値段の安さです。PROシリーズは構造が単純なので生産コストが安くすむはずです。新規購入や買い足し，買い替えにしても廉価版モデルがあれば，X68030も買いやすいってものです。5インチFDDで安いX68030を待っている人は多いのです。5インチFDD・HDDなしで20万円前後っていうのは無茶でしょうか。

私も含めて周りのX68000ユーザーはEXPERTシリーズとPROシリーズが全盛の頃に買った人がほとんどです。そして，その半数はPROシリーズを選んでいます。PROシリーズは明らかにX68000ユーザーの増加に一役買っていたのです。その辺のことをシャープは思い出してほしいなぁ。

中村　健（24）　X68000 ACE-HD，PC-386GS，AMIGA500，MSX2＋　埼玉県

●6歳の娘が「PUSH BON！」をえらく気に入りました。「お父さん，こいのぼりのでぃすくはどれ？」とディスクスタンドに立てておいたやつをぶちまけて聞いてきます。娘はひらがなしか読めないので，付録ディスクのラベルの「こいのぼり」しかわからなかったのです。しかたがないので展開した「PUSH BON！」に大きく「こいのぼり」と書きました。6歳の娘も知っているOh!Xの「こいのぼり」！

野原　賢次（33）　X68000 ACE-HD　埼玉県

●「ICE.R」ですが，PIC以来の快挙だと思う。PIC登場時のアルゴリズムの斬新さ（私が知らなかっただけかもしれない）には，とても驚かされた。あの稲妻の走り方……。そのPICの圧縮率を上回るICEの登場はさまざまなディスク媒体，通信媒体に影響を及ぼすと思う。

もうこれ以上の圧縮率は望めそうにないから，究極の画像圧縮ツールなのだろうか？

内藤　陽一（27）　X68000，PC-98NS/E　東京都

●新製品紹介「Workroom SX-68K/開発キット用ツール集」が大変参考になりました。

以前からSX用のプログラミングには興味があったのですが，いかんせん資料が少なくて，なにから始めればいいかわからない状態でした。この「Workroom SX-68K」も，対応が「SX-WINDOW ver2.0」までなので，どうしようかと思っていたところにこの記事ですので，買う決心がつきました。しかし，やはり高いので手に入るのはいつのことやら。

森崎　剛（21）　X68000 XVI，PC-9801RX21，FM TOWNSII HR20　広島県

●「知能機械概論」がとても興味深かったです。いままで人間の言語を調べて，そこから関連性を導き出したり，その変化の過程を調べるのが言語学だと思っていました。しかし，まったくの0から人間が仮想生物を創り出し，“プログラム”によって言語の情報処理過程を研究していくという考え方はすばらしいと思いました。ランギーの研究をぜひとも続けてほしいと思います。

八亀　桂一（19）　X68000 PRO　神奈川県

●「DōGA CGAアニメーション講座」の「アマチュアCGA学会」の内容は面白かった。しかし，それ以上にマンデル北尾君である。よく撮影を承知した（無理やり？）と思う。あそこまで顔をおもちゃにされるとかわいそうな気もするが，これもCGA発展のためと思って，これからもがんばってください。そのうちCGAマガジンで，DōGAメンバーの顔のマッピングデータ集が出たら面白いかな。

北風　保（22）　X68000 ACE　東京都

●横内さん，丹さんのエネルギーが切れそうであるとか。それはまずい。2人の目指す世界は，同時に我々ユーザーすべてが見たい世界でもあるので，ぜひともがんばっていただかなくては。確かに2人がはらわれている労力の割に，受け手側である我々からのレスポンスはあまりよくなかったかもしれない。うーん，技術的な面では協力もなかなか難しいし，こうやってエールを送るくらいしかできませんが応援しています。

橋本　和典（26）　X68000 XVI　東京都

ごめんなさいのコーナー

6月号　満開式磁帯駆動装置壱號

P.64　紹介されているストリーマの値段が間違っていました。正しくは外付けタイプの「MK-DSI」が138,000円で，内蔵タイプが125,000円（ともに税別）でした。ご迷惑をおかけしたことをお詫びいたします。

6月号　当代ハードディスク事情

P.57　同記事のコラム「小さいことは正義である・HDD編」において，「V2-2000」（通称NOVA2）の転送レートの上限が間違っていました。正しくは1Mバイト/秒です。ご迷惑をおかけしたことをお詫びいたします。

5月号　パズルゲームPUSH BON！

P.61　36面が解くことができませんでした。本当に申し訳ありませんでした。

5月号　EX_DES.X

P.58　MK_SORCE.Cにバグがありました。以下の部分を加筆，修正後，再コンパイルしてください。

```
149:誤  fprintf( fp, "void¥tback_ground( int i );¥t/* ID %2d */¥n", i );
149:正  fprintf( fp, "void¥tback_ground( int i );¥t/* ID %2d */¥n", i+1 );

211:誤  "¥t-1, -1, %d, %d, %d¥n};¥n¥n", win_x, win_y,ctrl_no+3
211:正  "¥t-1, -1, %d, %d, %d¥n};¥n¥n", win_x, win_y,ctrl_no+4

254:誤  i+2, i
254:正  i+3, i

288:誤  (ctrl[i].x1-ctrl[i].x0)/6,n
288:正  (ctrl[i].x1-ctrl[i].x0)/6,i

188～189の間に追加
fprintf( fp, ",¥n¥t 0,1,0,0,%3d,%3d,0,0,0,0", win_x,win_y );

249～250の間に追加
fprintf( fp, "¥t¥t¥tcase %2d:/* ID %2d */¥n¥t¥t¥t¥tbreak;¥n", i+2, i);
```

バグに関するお問い合わせは
☎03(5642)8182(直通)
月～金曜日16：00～18：00

お問い合わせは原則として，本誌のバグ情報のみに限らせていただきます。入力法，操作法などはマニュアルをよくお読みください。

また，よくアドベンチャーゲームの解答を求めるお電話をいただきますが，本誌ではいっさいお答えできません。ご了承ください。

聴こえてくるだろうか あのメロディが

▶いま音楽はありとあらゆるところに溢れています。もしも，そんな状況下で音楽がまったく聴けなくなってしまったら，あなたは耐えられるでしょうか？　想像さえもつかないかもしれません。それほど音楽は身近にあるのです。それに，音楽は人にいろんな影響を与えるような気がします。ある曲を聴くと元気になったり，人に対して温かい気持ちになったりするのです。

もちろん聴くだけが音楽ではありません。楽器を演奏したり，自分で作曲したり，音楽の楽しみ方は人それぞれです。ただ，好きな曲を聴くのは，とっても楽しいですよね。そして，好きな曲が自分の手によって作れるとしたら素晴らしいと思いませんか。

まず，その一歩として楽譜を読めるようにしたり，パソコンで音楽を鳴らしてみましょう。素晴らしい音楽の世界が，きっとあなたを待っていることでしょう。

▶今月は「LIVE in'94」でも，いつもよりページを多めにとって，短めの曲をたくさん用意してあります。ぜひ打ち込んで，聴いてみてくださいね。

▶さて，5月号で予告したとおり，第10期愛読者年間モニタ当選者の発表を行います(順不同，敬称略)。

矢野啓介(北海道)，中村俊之(東京都)，進藤慎一(青森県)，中矢史朗(愛媛県)，大上幸宏(鹿児島県)，野田類邦(愛知県)，渡辺祐介(富山県)，小林佳徳(新潟県)，壁谷善嗣(愛知県)，北野雅利(大阪府)，奥田直也(愛知県)，鈴木朝夫(神奈川県)，弦元達也(香川県)

以上13名の方にはこれから1年間，Oh!X愛読者年間モニタの一員として，がんばっていただきます。今月号から早速レポートを送りますので，よろしくお願いします。モニタにもれた皆さんも，たくさんの応募，本当にありがとうございました。

▶「ハードコア3Dエクスタシー」はページの都合によりお休みです。楽しみにしていた方には申し訳ありませんでした。

投稿応募要領

- 原稿には，住所・氏名・年齢・職業・連絡先電話番号・機種・使用言語・必要な周辺機器・マイコン歴を明記してください。
- プログラムを投稿される方は，詳しい内容の説明，利用法，できればフローチャート，変数表，メモリマップ（マシン語の場合）に，参考文献を明記し，プログラムをセーブしたフロッピーディスクを添えてお送りください。また，掲載にあたっては，編集上の都合により加筆修正させていただくことがありますのでご了承ください。
- ハードの製作などを投稿される方は，詳しい内容の説明のほかに回路図，部品表，できれば実体配線図も添えてください。編集室で検討のうえ，製作したハードが必要な場合はご連絡いたします。
- 投稿者のモラルとして，他誌との二重投稿，他機種用プログラムを単に移植したものは固くお断りいたします。

あて先

〒103　東京都中央区日本橋浜町3-42-3
ソフトバンク出版部
Oh!X「テーマ名」係

SHIFT・BREAK

▶私の部屋は電子レンジとセラミックファンヒーターを同時に動かすとブレーカーが落ちる。電源も以前は壁にコンセントがあったらしいのだが，そこは塞がれており，廊下から怪しい線が部屋に入っている。ちなみに廊下の電灯はよりによって私の部屋とブレーカーを共有していることもわかった。3段タコ足の先にあるX68000は大丈夫でしょうか？（H）

▶ニッパーがない！　おかしい。数時間前にはあったのに！　締め切り迫るなか3時間も探してしまった。考えられるところはすべて探したのにないから，神隠しにでもあったのか？　と翌日購入。買ってから出てくるのがマーフィーなんだよなと思ったものの，見事に布団の中から出てきたのはちょっと……。布団の中も探したはずだったのになぁ。（瀧）

▶歯磨きに凝っている。イオン効果で歯垢を落とす電子歯ブラシを手に入れたのがきっかけだ。こいつで磨くと歯がツルツルになって気持ちがいい。調子にのって糸つきようじと歯垢検知薬と電動歯ブラシで口内一斉大掃除。アイテム使うのが楽しいし，歯は健康になるし，いいことだらけと思ったら，埋めといた詰め物が立て続けに取れた。あらら。(E.K)

▶楽器フェアに行ってきた。目玉はヤマハのVP1。物理モデルで音をデザインするというまったくいままでのものとは違った視点で音を作るシンセだ。価格は270万円で4マルチ，16ポリ。1音につきWORKSTATION級のCPUが2個使われているというから1台で16×2＝32個のCPUが使われているそうな。いまにこの音源もパソコンに内蔵されるかな。（善）

▶マシン室にやってきたDOS/Vマシンに「World Circuit」をインストールする。英語モードに移行する手順を見つけるのに一苦労。そもそも英語モードという概念が存在する(多くの海外ソフトは英語モードでないと動かない)ことに愕然。いちいち腹の立つOSだ。しかしF-1マシンの滑らかな動きがすべてを免罪した。自分でも買っちゃおうかな。（A.T.）

▶国立競技場でグランパスとジュビロの試合を見た。VIP席にリネカーとファルカンがいて，人だかりがしていた。グランパスの辛勝だったが，高いところから見ると上手い下手が素人でもよくわかる。やはりテレビは肝心なところを映してくれない。それにしても90分の観戦は疲れる。じっとしたまま固い椅子で試合に集中。なかなか気持ちいいものだ。（K）

▶自律神経失調症で入院していた僕は，いまでは自宅療養中。まるで結核や鬱病の純文学者のよう，とは有田氏の弁。そういわれるとちょっとかっこいい。職場復帰は6月とも7月ともいわれているが，体調次第なので何ともいえない。入院中に起きた最大の社会情勢の変化はRがSになったことだろう。オープニングの歌手は前のほうが好きだな。（KO）

▶ある2頭の馬が放牧中に入れ代わってしまった。競馬場に戻ったそのうちの1頭が勝ち続けた。勝ち続けることで，周囲が疑問に思い始めた。そして，入れ代わったことがわかった。馬は馬主から数千万円で主催者に買い取られ，殺された。自分たちのミスを隠すために。馬に罪はなく，ただ一生懸命走っただけなのに。なんかやりきれない……。（高）

▶最近のOh!PCの読者アンケートの集計では，回答者のうち女性は2％だったそうだ。「えーっ，そんなに少ないのお」なんていったけど，Oh!Xはもっと少ない。読者ばかりかスタッフにも女性はほとんどいない(約4名)。社内のほかの編集部でも状況は似たりよったりだ。パソコンが「パーソナル」になるのは遠い未来のことみたいな気がしてきた。（ふ）

▶疑惑の「PUSH BON!」36面ですが，20～30ステップで解けるかも……というハガキを発見。本当に解けるのかなあ？　非常に気になりますが，解法を見つけた方，ハガキに書いて送ってください。なお，36面より先に進みたい人は，マップデータファイルを「PUSH-USR!.MAP」にして，適当にエディットしてくださいな。（K.Iさんいじめてごめんねの J）

▶本物のマインスイーパを初めてやった。簡単。こんな頭を使わないゲームとは思わなかった。ということで，話題のマル遅メディアを満喫する今日この頃。PSとサターンは買っちゃいそうだし，お，FXは。SCSIでつなげばCD-ROMドライブになるのか……。しかし今月は悪条件満載でちときつかった。やはり定期券の6の並びが悪かったのだろうか。（U）

▶響子さんに「パソコン入門者向けにやってもらいたいことがあるんだけど」と相談してみた。じつはOh!Xのことではなかったのだが，ふとOh!Xにはパソコン入門者と呼べる読者が果たしているのだろうかと気になってしまった。自分は入門者だと思う人がいたら，どうしてそう思うのか，ちょっと声を聞かせてほしい。アンケートハガキでね。（T）

microOdyssey

私の住むマンションの隣に，住宅にして3階建てほどの高さの，そう大きくはない建物が建った。倉庫か工場らしいが，工事中には「お騒がせしてすみません」の旨の挨拶状が来ていたし，ごく常識的に建設されていたので「ふぅん，なんか建つのね」という程度の感想だった。

ところが，いざ完成したものを見て，私は仰天した。なんのへんてつもない四角いものだが，問題はその色である。

初めは遠目に見て，工事中の網がかぶせてあるのだと思ったのだが，近づくにつれ，それは建物そのものの色であることが判明した。青みがかった濃い緑色。そうちょうど，1993年8月号のOh!Xの表紙の色をずっと濃くした感じのものだ。小さい建物とはいえ，その色は存在を激しく主張して，暑苦しくそこにある。

私は「自分のものをどんな色に塗ろうが自由だし」「見慣れれば気にならなくなるかも」とも思ったが，どうも慣れるのは難しいようだ。

普通は深夜に帰宅するので，夜の闇のなかで目立たない。しかし，徹夜明けに疲れて帰る昼間になど，日の光を浴びてけばけばと在るそれを見ないわけにはいかない。およそ周囲と調和しないその緑のカタマリを見ると，情けなさにへなへなとなってしまう。

周囲には，小さな印刷所や紙問屋，食堂などがあるが，普通の住宅地である。決して繁華街などではない。その建物は環境のなかにまったく融け込んでいないし，よい意味で人目を引いているわけではない。目立つ必要があるとも思えないから，どういう意図があるのか私にはわからないのだが，単にこの色が好きな人がいたとか，案外もっと単純にペンキがあったから塗ったという程度のことかもしれない。

と，まあ「実害」があるわけではないので，そんなつまらないこと，と思う人もいるかもしれない。しかし，果たしてどうでもいいことなのだろうか。自分ちの隣については個人的感情だといわれても仕方がないが，これ以外にも日本の街なかにはあちこちにこういうものがある。それらは見ていて決して愉快なものではない。

街の景観というのは，毎日なんの気なしに目にしているものである。最初は変だと思っても，知らず知らずのうちにそれが当たり前になり，判断力をなくしてしまうこともあるだろう。凝ったデザインがいいとか，シンプルでないと駄目だとかいうのではない。美しいものイコール善というのでもない。ただ，不調和はよくない，と思うのだ。

自宅の中なら他人には関係ないし，また，洋服などの色なら一時的なものだからいい。だけど，建築物はその外観だけでも環境を構成する要素のひとつである。決して個人だけの問題ではないはずだ。それを考慮せずにいると，感覚はだんだん麻痺していく。環境への影響を感じる力が鈍化するということは，ひいてはいろいろな意味で周りの対して配慮しないということにもつながっていくのではないだろうか。

環境のなかで，音や振動，臭いなどは「公害」としてはっきりと認識されている。建築物の高さや，敷地内の面積や建築場所などについても，法規で定められている。それなのになぜ，色については問題にならないのだろうか。人間の感覚への影響力は大きいと思うのだが……。(ふ)

1994年8月号7月18日(月)発売

特集　グラフィックあれこれ

・EX-WINDOW用外部ファイル
・SX-WINDOW用PICローダ
・SX-WINDOW用壁紙動画
・そのほかグラフィックに関するプログラムなど

新製品紹介
SX-WINDOW ver3.1/Mul-GS
全機種共通システム
シューティングゲーム作成講座(2)

バックナンバー常備店

東京　神保町　三省堂神田本店5F　03(3233)3312
〃　書泉ブックマートBI　03(3294)0011
〃　書泉グランデ5F　03(3295)0011
秋葉原　T-ZONE 7Fブックゾーン　03(3257)2660
八重洲　八重洲ブックセンター3F　03(3281)1811
新宿　紀伊国屋書店本店　03(3354)0131
高田馬場　未来堂書店　03(3209)0656
渋谷　大盛堂書店　03(3463)0511
池袋　旭屋書店池袋店　03(3986)0311
八王子　くまざわ書店八王子本店　0426(25)1201
神奈川　厚木　有隣堂厚木店　0462(23)4111
平塚　文教堂四の宮店　0463(54)2880
千葉　柏　新星堂カルチェ5　0471(64)8551
船橋　リブロ船橋店　0474(25)0111
〃　芳林堂書店津田沼店　0474(78)3737
千葉　多田屋千葉セントラルプラザ店　043(224)1333
埼玉　川越　黒田書店　0492(25)3138
川口　岩渕書店　0482(52)2190
茨城　水戸　川又書店駅前店　0292(31)0102
大阪　北区　旭屋書店本店　06(313)1191
都島区　駸々堂京橋店　06(353)2413
京都　中京区　オーム社書店　075(221)0280
愛知　名古屋　三省堂名古屋店　052(562)0077
〃　パソコンΣ上前津店　052(251)8334
刈谷　三洋堂書店刈谷店　0566(24)1134
長野　飯田　平安堂飯田店　0265(24)4545
北海道　室蘭　室蘭工業大学生協　0143(44)6060

定期購読のお知らせ

Oh!Xの定期購読をご希望の方は綴じ込みの振替用紙の「申込書」欄にある『新規』『継続』のいずれかに○をつけ，必要事項を明記のうえ，郵便局で購読料をお振り込みください。その際渡される半券は領収書になっていますので，大切に保管してください。なお，すでに定期購読をご利用の方には期限終了の少し前にご通知いたします。継続希望の方は，上記と同じ要領でお申し込みください。

基本的に，定期購読に関することは販売局で一括して行っています。住所変更など問題が生じた場合は，Oh!X編集部ではなくソフトバンク販売局へお問い合わせください。

海外送付ご希望の方へ

本誌の海外発送代理店，日本IPS(株)にお申し込みください。なお，購読料金は郵送方法，地域によって異なりますので，下記宛必ずお問い合わせください。

日本IPS株式会社
〒101 東京都千代田区飯田橋3-11-6
☎03(3238)0700

Oh!X 7月号
■1994年7月1日発行　特別定価850円(本体825円)
■発行人　橋本五郎
■編集人　稲葉俊夫
■発売元　ソフトバンク株式会社
■出版事業部　〒103 東京都中央区日本橋浜町3-42-3
Oh!X編集部　☎03(5642)8122
販売局　☎03(5642)8100 FAX 03(5641)3424
広告局　☎03(5642)8111
■印刷　凸版印刷株式会社

満開の電子ちゃん

作・え 岡村祭

76号以降は、毎月2枚組一、五〇〇円となります。73号(5/18発売)から半年の場合七、五〇〇円です。

講読方法：定期購読もしくはソフトベンダーTAKERUでお買い求めいただけます。
★定期購読の場合＝購読料第73号(94年6月号)より6ヶ月分7,500円(送料サービス、消費税込)を、現金書留または郵便振替で下記の宛先へお送り下さい。
現金書留の場合：〒171 東京都豊島区長崎1-28-23 Muse西池袋2F ㈱満開製作所
郵便振替の場合：東京 5－362847 ㈱満開製作所
●ご注文の際は、郵便番号・住所・氏名・電話番号を忘れずに記入して下さい。
●3.5インチディスク版をご希望の方は、「3.5インチ版」とご指定下さい。
●新規購読の方は「新規」と明記して下さい。なお、特に購読開始号のご指定がない場合は既刊の最新号からお送りいたします。
●製品の性格上返品には応じられませんが、お申し出があれば定期購読を解約し残金をお返しします。
★TAKERUでお求めの場合＝1部につき1,200円(消費税込)です。
●定期購読版と内容が一部異なる場合があります。御了承下さい。
●お問い合わせ先 TEL(03)3554－9282(月～金 午前11時～午後6時)
(なお、定期購読版のバックナンバーについては定期購読の方のみご注文を承ります)

澤田眞一(奈良県)

え～テキスト電脳倶楽部の119行目を見て。ほら、そこよそ見してるんじゃあない。いいかぁ、今日はSXerのための極悪win道からやっていくからな。近ごろじゃ、SX-WINDOW位いじれないとどこにも就職なんてできないぞ。何をごちゃごちゃいっている?えっ難しすぎる?先生はそんな子に教育した覚えはないぞ。仕方がない、君は113行目の初心者撲滅計画から読んでいきなさい。これなら十分に理解できるはずだ。次からはちゃんと予習してくるようにな。何、教科書がない。さっさと定期購読してこんか!!

「わんさかフェアー」開催中!

(6月18日～6月30日迄)

ツクモラジオCM.好評 ON AIR中!!
東京 ニッポン放送「伊集院光のoh!デカナイト」内
(月)～(木) PM10:30ごろ
名古屋 東海ラジオ 「大内ひろのしんの今夜もやっぱりFUNKYパジャマ」内
(月)～(金) PM11:00ごろ
札幌 FMノースウェーブ「SUPER CHOICE」
(月)～(金) PM7:30ごろ
ツクモイメージガール越智静香さんが茶目っ気たっぷりにパソコンをおねだり。
月末には特価品をズバリ!ツクモ特価でお知らせします!!
～これを聴いたら、ツクモに来ないではいられません～

TSUKUMO TSUKUMO TSUKUMO TSUKUMO TSUKUMO TSUKUMO

お申し込みは今すぐ! 受注専門フリーダイヤル **0120-377-999**

～ツクモ全店では"夏・ツクモ・ザ・バーゲン"開催中!ご来店、おまちしております!!～

ZAURUS ザウルス わんさかの目玉 (FAX&インク)新登場!

●出先でFAX OK!FAX発信システム付
PI-4000FX 定価¥91,000
●入力した自筆のまま見やすく表示。インクワープロ機能付
PI-4000 定価¥75,000
★フェアー中は特別価格にて御奉仕!!

本II・店頭デモスケジュール

上記、ザウルス→6/25,26,7/2,3
話題のカラーイメージスキャナー
JX-330シリーズ→6/18,19,25,26
※内容と日程は変わる場合もございます。
他にも良いモノたくさん!
わんさかに是非御来店下さい。

X680x0シリーズ本体

「わんさかフェアー」開催期間付き、特別セール実施中!!

なんと **69%OFF** です。

CZ-674C-H(X68000 CompactXVI)

超特価¥89,800(通常価格¥93,800)

TS-XFDCAを使えば、縦置き5インチモデルX68000シリーズ(PROシリーズを除く)を外部ドライブとして使用可能!
是非、2台目のマシンとしてどうぞ!
※モニタは別売です。

お勧めのセット

CZ-674C-H ………… **¥298,000**
CZ-607D-BK ………… **¥99,800**
RGBケーブル ………… サービス

ツクモ特価¥148,000

X68030のお勧めの組み合わせ!!

CZ-500C-B ………… **¥398,000**
240MBハードディスク ………… **サービス**

ツクモ特価¥298,000

満開製作所の商品も取扱中!

X68000 CompactXVI 24MHz改
RED ZONE ………… **ツクモ特価¥130,000**
RED ZONE(2DD) ………… **ツクモ特価¥135,000**

満開製外付け5インチFDD
MK-FD1 ………… **ツクモ特価¥39,800**
MK-FD1(カラーリングモデル) ………… **ツクモ特価¥44,800**

X68000/030シリーズ用RAMボード

型番	対応	ツクモ特価
SH-6BE1-1ME	(CZ-600C専用)	¥10,800
PIO-6BE1-AE	(ACE/PRO/PRO2シリーズ用)	¥10,800
PIO-6BE2-2ME	(拡張スロット用)	¥22,800
PIO-6BE4-4ME	(拡張スロット用)	¥38,800
SH-5BE4-8M	(X68030シリーズ用)	¥44,800
CZ-6BE2A	(XVI専用)	¥42,500
CZ-6BE2D	(CompactXVI専用)	¥29,800
TS-6BE2B	(CZ-6BE2A/D用拡張RAM)	¥29,800
TS-XM1-4A	(拡張スロット用4MB)	¥39,800
TS-XM1-6A	(拡張スロット用6MB)	¥47,800
TS-XM1-8A	(拡張スロット用8MB)	¥55,800
TS-XM1-10	(拡張スロット用10MB)	¥63,800
X SIMM10-8M	(拡張スロット用8MB)	¥53,800
X SIMM10-10M	(拡張スロット用10MB)	¥64,800

TS-3XRシリーズ

X680x0用外付けドライブ
・2DD/2HD/2HC/1.44MBフォーマット対応
※2DD/2HC/1.44MBを使用するにはHuman68K Ver.3.0以上が必要
・CompactXVI/68030用ケーブル付

TS-3XR1B 1ドライブ 定価¥33,800 ………… **ツクモ特価¥26,800**
TS-3XR2B 2ドライブ 定価¥46,800 ………… **ツクモ特価¥36,800**

ツクモオリジナル ジョイスティックパラレルインターフェイス

TS-JPIFSforCZ-8NS1

近日発売予定

定価¥17,800
発売記念特価¥14,800

拡張スロットを使用しません。ジョイスティック端子に接続できるパラレルインターフェイスです。これでスキャナーも高速で取り込みが可能になります。取り込みソフトェア及びサンプルソースが付属致します。

ディスプレイも特別価格にて提供中!

CZ-607D(14型カラーディスプレイテレビ)
ツクモ特価¥60,000

CZ-608D(14型カラーディスプレイ)
ツクモ特価¥69,000

CZ-615D(15型カラーディスプレイテレビ)
ツクモ特価¥132,000

CZ-621D(21型カラーディスプレイ)
ツクモ特価¥125,000

プリンター

48ドットカラー熱転写プリンター
CZ-8PC5-BK **限定特価¥39,800**

バブルジェットプリンター
BJ-10VLite(ケーブルセット) **ツクモ特価¥38,500**

カラーバブルジェットプリンター
BJC-600J(ケーブルセット)
ツクモ特価¥94,800

MIDIインターフェース

CZ-6BM1A ………… **ツクモ特価¥19,000**

マッハジェットカラー 6月下旬発売予定!!
MJ-700V2C(ケーブルセット)
ツクモ特価¥83,800

カラーイメージスキャナー

JX-330X NEW

7月発売予定!予約受付中
定価¥178,000
高解像度(600dpi)、超高速が特長です。
ADF・透過原稿対応型カラーイメージスキャナの登場です。
Scanner Tool/s(画像入力ソフト)付属。
ツクモ特価¥138,000

CZ-8NS1 **台数限定**
ツクモ特価¥69,800

【東 京】●パソコン本店(各種パソコン・周辺機器)●パソコン本店II(パソコン・ワープロ)●DOS/Vパソコン館(DOS/Vパソコン・下取り)●万世店(総合通信機器)●5号店(ビデオ・ムービー・CS)●ソフト8号店(パソコン&ゲーム用ソフト)●買取センター(ゲーム機・ゲーム機用ソフト買取り)●ニューセンター店(パソコン・中古・下取り・買取り)●ラジオセンター店(ハンディーレシーバー・テレホンパーツ)
【名古屋】●名古屋1号店(パソコン全般)●名古屋2号店(パソコン全般・総合通信機器・ビデオ) 【札 幌】●札幌店(パソコン全般・総合通信機器)●DEPOツクモ札幌(パソコン全般)

ズバリご奉仕

P&Aならではの 新品パソコン5年保証

《業界No.1の"P&Aメンテナンスサポート"》

最高の保証システム

①業界最長の新品パソコン5年保証
（※モニター・プリンター3年間保証.// ※一部商品は除きます。）
②中古パソコンの1年間保証（※モニター・プリンター6ヶ月間保証.//）
③初期不良交換期間3ヶ月（※新品商品に限らせていただきます。）
④永久買取保証
⑤配達日の指定OK//（土曜・日曜・祭日もOK.//）
⑥夜間配達もOK//（※PM6:00～PM8:00の間 ※一部地域は除きます。）

便利でお得な支払いシステム

①翌月一括払い手数料無料（ご利用下さい。）
②業界No.1の低金利//
③月々の支払いは¥1,000より
④9ヶ月先からのスキップ払いOK//
⑤84回までの分割、ボーナス併用OK//
⑥カレッジクレジット
⑦ステップアップクレジット
⑧ボーナスだけで10回払いOK//
⑨現金一括支払いOK//
⑩商品到着払いOK//（代引き手数料が必要になります。10万円まで900円）
（※商品・金額ご確認の上、銀行振込・現金書留にてご入金下さい。）

●法人向け リースシステム
業務に最適なシステムを構築します。
損金処理が可能なリース契約をどうぞ。

周辺機器コーナー （送料¥1,000・消費税別）

カラーイメージスキャナ
■JX-325X
定価¥190,000
特価¥99,800

カラーイメージジェット

■IO-735X-B
定価¥248,000
特価¥128,000

ビデオスキャナー
■CZ-6VS1
定価¥178,000
特価¥135,000

FDD（5インチ×2基）
■CZ-6FD5
定価¥99,800
P&A超特価
¥49,800

プリンター（ケーブル用紙付）
- ●MJ-500V2 （エプソン）……特価¥48,500
- ●MJ-1000V2（ 〃 ）……特価¥71,300
- ●VP-1047PC（ 〃 ）……特価¥49,000
- ●BJ-220JC （キヤノン）……特価¥58,000
- ●BJ-10V Lite（ 〃 ）……特価¥34,500
- ●BJ-15V PRO（ 〃 ）……特価¥46,500
- ●LBP-A404GII（ 〃 ）……特価¥99,500
- ●BJC-820J （ 〃 ）……特価¥154,300
- ●JET505J PLUS（YHP）……特価¥53,300

光磁気ディスク（X68000用）
■CS-M120（コパル）
●ケーブル、ターミネータ付
¥178,000
特価¥96,500

- ●CZ-6BV1………定価¥21,000▶特価¥15,900
- ●CZ-8NM3………定価¥ 9,800▶特価¥ 7,200
- ●SH-6BF1………定価¥49,800▶特価¥36,500
- ●CZ-6BP1………定価¥79,800▶特価¥57,000
- ●CZ-6BS1………定価¥29,800▶特価¥21,500
- ●CZ-8NJ2（限定）…定価¥23,800▶特価¥13,800
- ●CZ-6CS1（674C用）・定価¥12,000▶特価¥ 8,900
- ●CZ-6CR1（RGBケーブル）・定価¥ 4,500▶特価¥ 3,600
- ●CZ6CT1（テレビコントロール）・定価¥ 5,500▶特価¥ 4,400
- ●CZ-6BP2………定価¥45,800▶特価¥33,300
- ●CZ-5MP1（X68030用）・定価¥54,800▶特価¥42,000

送料¥700・消費税別

■システムサコムボード
●SX-68MII（MIDI）
定価¥19,800
特価¥13,500
●SX-68SC（SCSI）
定価¥26,800
特価¥17,500

X68000用ソフトコーナー （送料¥700・消費税別）

- ●Z's STAFF PRO68K Ver.3.0（ツァイト）……定価¥58,000▶特価¥37,500
- ●Z's TRIPHONYデジタルクラフト（ツァイト）……定価¥39,800▶特価¥27,000
- ●マジックパレット（ミュージカルプラン）……定価¥19,800▶特価¥14,200
- ●たーみのる2（SPS）……定価¥17,800▶特価¥13,000
- ●Mu-1 Super（サンワード）……定価¥39,800▶特価¥28,500
- ●サイクロンEXPRESS α68……定価¥98,000▶特価¥69,000
- ●Video PC for X680X0（マイクロウェアシステムズ）……定価¥58,000▶特価¥46,400
- ●X WINDOWS V.11.5（マイクロウェアシステムズ）……定価¥30,000▶特価¥25,500
- ●Double Book IN（計測技研）……定価¥12,800▶特価¥ 9,600
- ●OS-9/X68030 V.2.4.5（マイクロウェアシステムズ）……定価¥25,000▶特価¥19,900
- ●C&Professional Pack V.3.2（マイクロウェアジャパン）……定価¥80,000▶特価¥57,800
- ●マチエール Ver.2.0……定価¥39,800▶特価¥28,800
- ●CZ-213MSD MUSIC PRO68K……定価¥18,800▶特価¥13,200
- ●CZ-214MSD SOUND PRO68K……定価¥15,800▶特価¥11,300
- ●CZ-215MSD Sampling PRO68K……定価¥17,800▶特価¥12,500
- ●CZ-220BSD DATA PRO68K……定価¥58,000▶特価¥40,000
- ●CZ-225BSV Multiword Ver.2.0……定価¥32,000▶特価¥23,000
- ●CZ-243BSD CYBERNOTE PRO68K……定価¥19,800▶特価¥15,000
- ●CZ-247MSD MUSIC PRO68K（MIDI）……定価¥28,800▶特価¥20,500
- ●CZ-249GSD CANVAS PRO68K……定価¥29,800▶特価¥22,000

☆ゲームソフト25%OFF OK.//（一部ソフト除く）

- ●CZ-251BSD Hyperword……定価¥39,800▶特価¥29,400
- ●CZ-253BSD CARD PRO68K Ver.2.0……定価¥29,800▶特価¥22,700
- ●CZ-257CSD Communication PRO68K Ver.2.0……定価¥19,800▶特価¥15,300
- ●CZ-258BSD Teleportion PRO68K……定価¥22,800▶特価¥16,900
- ●CZ-261MSD MUSICstudio PRO68K Ver.2.0……定価¥28,800▶特価¥21,200
- ●CZ-263GWD Easypaint SX-68K……定価¥12,800▶特価¥ 9,800
- ●CZ-264GWD Easydraw SX-68K……定価¥19,800▶特価¥15,300
- ●CZ-265HSD NewPrint Shop Ver.2.0……定価¥20,000▶特価¥15,400
- ●CZ-266BSD PressConductor PRO68K……定価¥28,800▶特価¥22,000
- ●CZ-267BSD CHART PRO68K……定価¥38,000▶特価¥29,800
- ●CZ-271BWD EG-Word……定価¥59,800▶特価¥44,900
- ●CZ-272CWD Communication SX68K……定価¥19,800▶特価¥14,500
- ●CZ-274MWD MUSIC SX68……定価¥38,000▶特価¥29,300
- ●CZ-275MWD SOUND SX68K……定価¥15,800▶特価¥11,500
- ●CZ-284SSD OS-9/X68000 Ver.2.4……定価¥35,800▶特価¥25,600
- ●CZ-286BSD BUSINESS PRO68K……定価¥28,000▶特価¥20,500
- ●CZ-288LWD開発キット（workroom）……定価¥39,800▶特価¥29,700
- ●CZ-289TWD 開発キット用ツール集……定価¥12,800▶特価¥ 9,600
- ●CZ-290TWD SX-WINDOW ディスクアクセサリー集……定価¥14,800▶特価¥11,500
- ●CZ-296SS（5"）/SSC（3.5"）SX-WINDOW Ver.3.1……定価¥22,800▶特価¥17,600
- ●CZ-295LSD C-Compiler PRO68K Ver.2.1 NEW KIT……定価¥44,800▶特価¥32,500

P&A 株式会社ピー・アンド・エー

〒124 東京都葛飾区新小岩2丁目2番地20号
●営業時間：AM10：00～PM7：00 日・祭：AM10：00～PM6：00
☎03-3651-0148（代）
FAX.03-3651-0141 MAC/DOS Vフロア ☎03-3655-4454
●定休日/毎週水曜日

全国通販

★頭金なし！
★即日発送

●お近くの方はお立寄り下さい。専門係員が説明いたします。
●本体単品で特価で受付します。詳しくは電話にてお問合せ下さい。
●ビジネスソフト定価の20%引きOK！TELください。

P&A特選 今月の中古特選品

- ●CZ-600C‥¥45,000
- ●CZ-601C‥¥55,000
- ●CZ-611C‥¥60,000
- ●CZ-652C‥¥65,000
- ●CZ-612C‥¥85,000
- ●CZ-603C‥¥75,000
- ●CZ-653C‥¥68,000
- ●CZ-612C‥¥80,000
- ●CZ-623C‥¥80,000
- ●CZ-674C‥¥80,000
- ●CZ-634C‥¥110,000
- ●CZ-644C‥¥155,000

※上記は単品価格、モニター別売。

新古品 限定
●CZ-674CH
●CZ-608DH
¥138,000
中古品
●CZ-674CH
●68000専用モニター付
¥99,000

限定
●CZ-634CTN（チタン）（中古）
●CZ-613D（グレー）（新品）
¥190,000
（モニターをCZ-614TN（チタン）に変更の場合¥20,000加算）
中古品
●CZ-623C-TN
●68000専用モニター
¥98,000

新古品 限定
●CZ-644CTN
●CZ-604DB
¥228,000
中古品
●CZ-644CTN
●68000専用モニター付
¥198,000

高額買取り（新品もOK）格安販売

■まずはお電話下さい。
下取り専用 買取り電話▶ ☎03-3651-1884 FAX.03-3651-0141
■下取り・買取りで、お急ぎの方は、直接当社に来店、または宅急便にてお送りください。

買取り価格…完動品・箱／マニュアル／付属品の価格です。中古販売…1年間保証付。

●下取りの場合…価格は常に変動していますので査定額を電話で確認してください。（差額は、P&A超低金利クレジットをご利用ください。）
●買取りの場合…現品が着き次第、3日以内に高価買取金額を連絡し、振込み、又は書留でお送り致します。
●近郊の方はP&A本店に直接お持ちください。即金にて¥5,000,000までお支払い致します。

●最新の在庫情報・価格はお電話にてお問い合せください。
●買い取りのみ、または、中古品どうしの交換も致します。詳しくは電話にて、お問い合せください。
●価格は変動する場合もございますので、ご注文の際には必ず在庫をご確認ください。
●本商品の掲載の商品の価格については、消費税は、含まれておりません。
●現金書留及び銀行振込でお申し込みの方は、上記商品の料金に3%加算の上でお申し込み下さい。詳しくは、お電話でお問い合せください。

P&A特選 パソコンラック＆OAチェアー（消費税込み）（送料無料、離島を除く）

①3段¥8,240 ②4段¥9,785
③4段¥12,875（持ち帰り可能です ご来店下さい。）

※全機種→キャスター付 ※フレーム色：ホワイト
※上から2番目棚板移動可能（4段）
※3段の場合、上から2番目の棚板は付いておりません。

※上下2分割式/スライドマウステーブル、中棚板は2段階に可動します。
※フレーム色：グレー

①¥9,270
●布張り ダークグレー
●ガスシリンダー

②¥11,330
●布張り ダークグレー
●ガスシリンダー
●肘付

通信販売お申し込みのご案内

〔現金一括でお申し込みの方〕
●商品名およびお客様の住所・氏名・電話番号をご記入の上、代金を当社まで現金書留でお送りください。（プリンター・フロッピーの場合、本体使用機種名を明記のこと）
〔クレジットでお申し込みの方〕
●電話にてお申し込みください。クレジット申し込み用紙をお送りいたしますので、ご記入の上、当社までお送りください。●現金特別価格でクレジットが利用できます。残金のみに金利がかかります。●1回～84回払いまで出来ます。但し、1回のお支払い額は¥1,000円以上。
〔銀行振込でお申し込みの方〕
●銀行振込ご希望の方は必ずお振込みの前にお電話にてお客様のご住所・お名前・商品名等をお知らせください。（電信扱いでお振込み下さい。）

〔振込先〕さくら銀行 新小岩支店
当座預金 2408626 (株)ピー・アンド・エー

超低金利クレジット率

回数	3	6	10	12	15	24	36	48	60	72
手数料	2.6	3.5	4.4	4.9	7.8	10.4	14.4	18.9	24.4	31.8

※お支払いは、便利な商品到着払い〈手数料（10万円まで900円）要〉をご利用下さい。

●価格は変動します。ご注文の際は必ずお電話で価格と在庫をご確認下さい。●本広告に掲載の商品には送料及び消費税は含まれておりません。

COMPUTER 恋LAND 夢LAND

東京ゲームデザイナー学院

PHONE 03-3370-2720

〒151 東京都渋谷区代々木3-55-28
資料請求は、お気軽にお電話下さい。(無料)

Rec

Rew

Play

Few

Still

Stop

入学資格は特にありません。

ゲームデザイナー養成講座コース一覧

	コース	曜日・時間	内容
全日制	1年コース	月〜金曜日 AM10:00〜PM4:00	1年間でゲームのデザインからゲームプログラムの制作までの、ゲーム制作の一連の流れを全てマスターするコースです。
全日制	2年コース	月〜金曜日 AM10:00〜PM4:00	ゲームデザインからプログラム制作までを2年間かけてじっくりと勉強できます。時間がありますから凝ったコンピューターゲームを制作することができます。
単科	ゲームデザイン	火・金曜日 午前 AM10:00〜PM12:30 午後 PM 6:45〜PM 9:00	ゲームの企画・プランニングを勉強するためのコースです。
単科	ゲームシナリオ	火・金曜日 午前 AM10:00〜PM12:30 午後 PM 6:45〜PM 9:00	コンピューターゲームのシナリオの研究・制作を中心に勉強するコースです。
単科	ゲームCG	月・木曜日 or 火・金曜日 午前 AM10:00〜PM12:30 午後 PM 6:45〜PM 9:00	ゲームに用いられるグラフィックプログラミングを勉強するコースです。コンピューターゲームのグラフィックデザイナーを志す方に薦めます。
単科	ゲームキャラクターデザイン	月・木曜日 or 火・金曜日 午前 AM10:00〜PM12:30 午後 PM 6:45〜PM 9:00	ゲームキャラクターのデザイナーを目指す方のコース。キャラクターエディターを使うコースと、プログラミングによるキャラクターの作成のコースがあります。
単科	ゲームサウンド	月・木曜日 or 火・金曜日 午前 AM10:00〜PM12:30 午後 PM 6:45〜PM 9:00	コンピューターゲームに用いられるサウンドプログラミングを勉強するコースです。主としてサウンドドライバーを作成するコースです。
単科	ゲームプログラミング	月・木曜日 or 火・金曜日 午前 AM10:00〜PM12:30 午後 PM 6:45〜PM 9:00	コンピューターゲームを題材にしながら、C言語又はアセンブラによる実践的なゲームプログラミングを中心に勉強するコースです。
単科	ゲームデザイナー	月・木曜日 or 火・金曜日 午前 AM10:00〜PM12:30 午後 PM 6:45〜PM 9:00	勉強時間があまり採れない人を対象にコンピューターゲームの企画から、様々なゲームの制作の流れをマスターするコースです。

※単科コースについては土曜日週一回コースも設定されています。
単科コースの期間設定は基本的に6ヶ月ですが初心者と経験者の違いによって、期間設定を変えてあります。詳しいことは、パンフレット請求の上、お確かめ下さい。

通信教育講座募集中！

当学院ではお忙しい学生や社会人及び通学出来ない方のために、各種通信講座を用意しておりますので、どうぞ御利用下さい。

［通信講座の主なコース］

《初心者コース》

■プログラミングの経験の無い方向け
Aコース

■BASICをマスターした方向け
Bコース

■プログラミングの経験の無い方向け
Cコース

《経験者コース》

■BASICゲームプログラミングコース
■C言語ゲームプログラミングコース
■アッセンブラゲームプログラミングコース
■ゲームデザイナーコース

《X-68000用ゲームCGコース》

■キャラクター&プログラミングコース
■キャラクターツールコース

《PC-9801用CGコース》

■グラフィックツールコース

受講期間は、6ケ月から2年までになっておりますので詳細は当学院までお電話でパンフレットを御請求下さい。

■6月、7月生募集！(通学)

◇**全日制1年、2年コース**
月曜日〜金曜日

◇**単科コース**
月曜日と木曜日又は、
火曜日と金曜日の週2回又は、
土曜日の週1回

94年10月生　願書受付中!

全日制1年、2年コース(月曜日〜金曜日) **単科**(週2、週1)
只今、学校見学会を月曜日〜金曜日 (PM1:00〜PM7:00)
に実施しております。ご好評いただいておりますので、お気軽にお越し下さい。

なお、御来校される時は、あらかじめお電話を下さい。

1995年4月生 学校見学会実施中！

ソフトバンクの新刊・近刊

EXCEL Ver.5 FirstBook

●エクスメディア 著
●予価2,900円

Excelはすでに MS-WINDOWS上では必備のソフトとなりました。Ver.5と進化し、この1本でワープロの仕事以外は全てこなせます。その外観を知る一冊です。

'94すぐに使えるオールインワンパソコン選択ガイド

●PC MOOK編集部 編
●定価1,500円

本書はユーザーがオールインワンパソコンを購入するにあたって抱く疑問や、知っていると得する情報を満載。また、実際に代表的なオールインワンパソコンを並べて紹介し、あなたの目的とそれを実現する待望のマシンの選択法をわかりやすく紹介します。

日本語Windows 1994年度版 オンラインソフトウェア集 ②

●NIFTY-Serve WINDOWSフォーラム 監修
●定価3,200円

NIFTY-Serve WINDOWSフォーラム主催のWINDOWS AWARDで、優秀な成績をおさめたオンラインソフトを一同に集めました。ファイル管理プログラムからエディタ、通信ソフトまで「使えるソフト」満載です。

C言語プログラミングレッスン 入門編

●結城浩 著
●予価2,400円

MS-DOSユーザとUNIXユーザを対象に、わかりやすさと正確さに徹底的にこだわった新しいタイプのC言語の入門書。初めてCを学ぶ初心者から、Cに挫折した経験者まで、Cユーザにとって最適の入門書です。

NetWare管理（上）導入編

●ビル・ローレンス 著/佐藤茂良 監訳/柴崎実 訳
●定価4,500円

本書は"何故なのか"と説明することで、コマンドやユーティリティも目的と現実的な応用法についての理解を促す構成になっています。ネットワーク管理者のみならず、これからネットワークをインストールしなければならない人、NetWare以外のことを知りたい人にもお薦めの一冊です。

あなたもフリーソフト作者になれるかもしれない

●石川英明 著
●予価2,700円

世の中、フリーソフト流行のご時世。人の作ったものをただ使うばかりではなんとも面白みがない。本書はどうしたら作る側にまわれるのかを楽しく解説しました。

Far Roads to Lordリプレイ RPGセッションガイド

●遊演体 監修/司史生・ゆうせぶん 共著
●予価1,600円

国産テーブルトークRPGシリーズの最新作、「Far Roads to Lord」の製作にあたったスタッフによる、初の公式ガイドブック。初心者から上級者まで、すべてのテーブルトークRPGファンにも読み物として楽しめる内容です。

スーパーストリートファイターII スーパーガイド

●Theスーパーファミコン編集部特別編集
●予価未定

格闘アクションゲームの人気シリーズ最新作の完全攻略本。新たに4人を加えた全16人の登場キャラクター別に、対戦攻略の必勝法を伝授。"スーパーストII"で友達に差をつけたい人への、徹底マニュアルガイド。

SIMCITY2000 パーフェクトガイド

●中島理彦 著
●定価1,600円

プレイヤーが市長となり、自分だけの都市作りが自由に楽しめる大人気シミュレーションゲーム「SIMCITY2000」の完全攻略ガイド。著者自身が作り上げたオリジナルの東京とパリの製作過程や困った時に役立つ目的別INDEXも収録。Mac版、DOS/V版、PC-98版すべてに対応。これであなたも立派な市長さんになれる!?。

第2種受験対策
速習ソフトウェア

●中山恵介 著
●予価2,200円

新しい試験制度のソフトウェア分野の必須テーマを、効率的に学べるように解説。図表を見て理解できる構成なので、どんどん頭に入り、弱点だったソフトウェアもこの一冊で完璧です。

第2種受験対策
速習 C言語

●中山恵介 著
●予価2,200円

Cの基礎から応用までを、効率的に学べるように解説。アルゴリズムを通して概要、次に詳細に基礎、最後に実践問題で応用、とステップアップ式で学べるので、確実に実力が身に付く内容です。

定価は税込です
お近くの書店でお求めください

SOFT BANK ソフトバンク株式会社/出版事業部

感性を光らせる。

さまざまなフィールドで、研ぎ澄まされた感性に応える潜在能力の実証

X68の潜在能力は、まさに時代とともに証明されつつあります。
開発当初より、現在のマルチメディア環境を想定していた事実。
グラフィック能力はもちろん、ADPCM対応、オリジナルウィンドウシステム、
X68にとってこれらは、数年前のスペックなのです。
パソコンの存在そのものを革新した「創造性」、マインドを喚起する「こだわり」、
いま、先見のユーザーに支えられたX68は
そのコンセプトの開花を得て、多彩なフィールドへと飛翔します。

Workbench
WSとしての楽しみ

たとえば、リアルタイム・マルチタスク・オペレーティング・システムOS/9。
X68030の能力を最大限に引き出すUNIXライクな操作性と洗練された機能。
X-WINDOWや動画ツールのサポートでさらに深い楽しみが…。

※OS/9はマイクロウェア・システムズ㈱の登録商標です。
※UNIXは、X/Openカンパニーリミテッドが独占的にライセンスする米国および他の国における登録商標です。

Create
創造するよろこび

SX-WINDOW開発支援ツールが
創造力を刺激する。
ソフト開発に必要なツールや
サンプルプログラムを多彩にバンドル、
ウィンドウ上で効率よく作業でき、
初めてプログラムに挑む人への
やさしい配慮が、創造するよろこびを
さらに高めてくれるでしょう。

Ammusement
遊びへのこだわり

X68の能力の高さを端的に示す
アミューズメントフィールド。
マインドをきわめたゲームフリークの
熱い期待に応える。
画像の美しさが感性を刺激する、
たとえばひと味違う大魔界村なら、
キミのこだわり度は今、全開!

X68030 / X68000
32bit PERSONAL WORKSTATION / PERSONAL WORKSTATION ·XVI

X68030 [本体+キーボード+マウス・トラックボール]
130㎜FD(5.25型)タイプ CZ-500C-B(チタンブラック) 標準価格398,000円(税別)・〈HD内蔵〉CZ-510C-B(チタンブラック) 標準価格488,000円(税別)

X68030 Compact [本体+キーボード+マウス]
90㎜FD(3.5型)タイプ CZ-300C-B(チタンブラック) 標準価格388,000円(税別)・〈HD内蔵〉CZ-310C-B(チタンブラック) 標準価格478,000円(税別)

X68000 XVI Compact [本体+キーボード+マウス]
90㎜FD(3.5型)タイプ CZ-674C-H(グレー) 標準価格298,000円(税別)

●ディスプレイは別売です。●消費税及び配送・設置・付帯工事費、使用済み商品の引き取り費等は、標準価格には含まれておりません。●画面はハメコミ合成です。

■お問い合わせは… シャープ株式会社 電子機器事業本部システム機器営業部 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)

T1002179070852 雑誌 02179-7